Texture Classification using Combination of LBP and GLRLM Features along with KNN and Multiclass SVM Classification

Sourajit Das
Dept. of Electronics & Telecommunication Engineering,
Veer Surendra Sai University of Technology,
Sambalpur, India
E-mail Id: sourajit19@gmail.com

Uma Ranjan Jena
Dept. of Electronics & Telecommunication Engineering,
Veer Surendra Sai University of Technology,
Sambalpur, India
E-mail Id: urjena@rediffmail.com

***Abstract*—The paper presents a unique combination of texture feature extraction techniques which can be used in image texture analysis. Setting the prime objective of classifying different texture images, the Local Binary Pattern (LBP) and a modified form of Gray Level Run Length Matrix (GLRLM) are implemented initially. The next phase involves use of combination of the former two methods to extract improved features. The feature vectors were obtained by defining the features on the transformed images. These texture features are classified using two classification algorithms, KNN and multiclass SVM. The results of above feature extraction techniques with individual classifiers have been compared. The comparison yields that the combination of LBP and GLRLM texture features shows better classification rate than the features obtained from individual feature extraction techniques. Among the classifiers, Support Vector Machine has better classification rate than the Nearest Neighbor approach for the texture classification.**

Keywords: Image Texture Analysis, Classification Algorithm, Feature Extraction, Nearest Neighbor, Support Vector Machine

I. INTRODUCTION

In Computer Vision, the ultimate aim is to use computers to imitate human vision, which includes learning and being able to make conclusions and take decisions according to visual inputs. It tries to do what a human brain does with the data provided by vision/sensory organs (eyes) i.e. understanding the issue based on image data. Texture represents the characteristic appearance of the surface of an object attributed by the size, shape, arrangement, depth of the constituent elements. A texture image is usually described in terms of smoothness, softness, hardness, or whether it is coarse or fine, matt or glossy etc. In short Texture Classification is the problem of distinguishing between different textures [1], [2]. Thus the objective of the given technique is to assign any unknown or test image to one of the set of previously known texture classes. From the previous works, it is found that most of the machine vision based texture classification techniques have combined texture features with classifiers to produce reasonably good classification accuracy for different images [3]-[6]. The microscopic images of the texture images provide better information content to precisely classify the variety of samples as compared to the macroscopic images which disclose restricted amount of information. Some of the popular techniques involved the use of Gabor filters, LBP, Image Pyramids, GLCM GLRLM, etc. LBP is an easy, yet computationally efficient algorithm used by many researchers whereas GLRLM has gained popularity in the recent years for applications in image processing and computer vision.

In this paper, a slightly modified form of the GLRLM based texture feature extraction method in addition to the traditional LBP, for microscopic images of textures has been proposed. Another technique combining both the LBP and GLRLM has also been proposed for better classification. The KNN and multiclass SVM classifier is employed to classify the different image textures. The effectiveness of the texture feature extraction techniques on the texture images was investigated for both the classifiers. The comprehensive review of LBP and GLRLM features, the KNN and SVM classifiers is described in the succeeding sections followed by a thorough discussion on the experimental results and the performance estimation of the proposed method.

II. METHODOLOGY

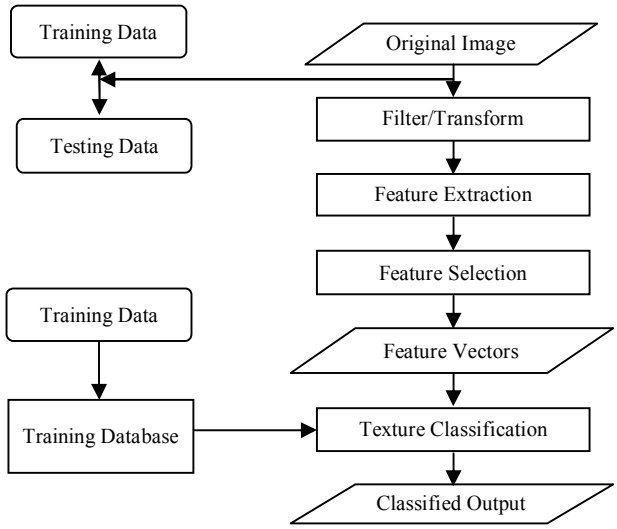

Fig. 1: Steps of Texture Classification

978-1-5090-3210-5/16/$31.00 © 2016 IEEE

The first step of texture classification starts with dividing the macroscopic texture images of the training and testing set into microscopic subimages. A 128x128 subimage is to be obtained from 512x512 texture image of the Brodatz database. The subimages were obtained starting from the first pixel and then shifting the mask by 32 pixels in both x and y direction resulting in 144 overlapped subimages for every single image in the database. These 144 subimages were divided among the training and testing datasets. The following steps of subimage transformation till the classification of image data are shown in the flowchart of Fig. 1.

A. Feature Extraction using Local Binary Pattern (LBP)

Local Binary Pattern (LBP) is a typical feature descriptor used for texture classification in image processing. Since its introduction by Ojala [7] in 1994, LBP has been observed to be a highly efficient feature descriptor for texture classification. A 3x3 mask is used against the neighborhood pixels to define a particular texture and evaluate a local binary pattern (LBP). LBP can be said to be the first-order circular derivative of the micro patterns present in an image which can be constructed by conjugating all the binary gradient directions. The conventional LBP operator described by Ojala produces information that is undeviating from the local grayscale randomness in the image [8]. It is computed for every pixel, taking the gray level intensities of a circular neighborhood (having radius R pixels) surrounding the central pixel q_c. Mathematically, the LBP operator can be defined as follows:

$$LBP(P,R)=\sum_{p=0}^{p-1} s(q_p - q_c)2^p \qquad (1)$$

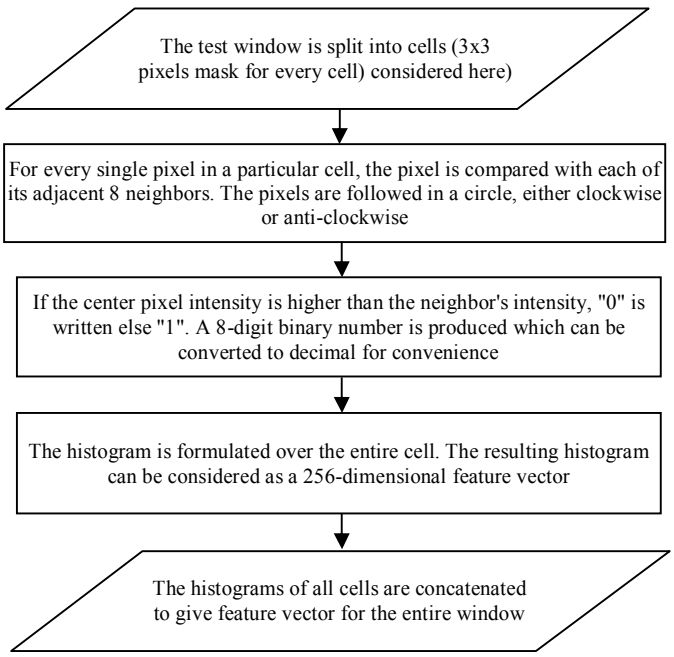

Fig. 2: LBP Applied on a Single Subimage using a 3 × 3 Mask

where P indicates the number of pixels in the neighborhood, and s(x) = 1 when x ≥ 0, else 0. Now a histogram of the obtained binary numbers is computed to represent the texture of the image. Observing the Fig. 2 below, the simplest LBP feature vector is created by the following steps:

The feature vector is then taken up for processing using the Nearest Neighbor and Support vector machine algorithm to classify the texture images. Applications such as face recognition or Pattern recognition also employ such classifiers.

B. Feature Extraction using Gray Level Run Length Matrix (GLRLM)

Considering the area of statistical texture analysis, texture features are mainly extracted using the statistical distribution of different combinations of gray level values at particular positions inside the image periphery. The Gray Level Run Length Matrix (GLRLM) developed by Galloway in 1975, paved the way to extract higher order statistical texture features [9]. A set of sequential pixels having the same gray level intensity and collinearity in a given direction, form a gray level run. The number of pixels present in the run defines the run length. Similarly the run length value indicates the number of occurrences of such a run in the image. These Run Length matrices highlight the roughness of a texture in a given direction. Further the GLRLM is generally a two-dimensional matrix in which every element $P(i,j|\theta)$ indicates the total number of occurrences of runs of length j at gray level i, in a particular direction θ. For example suppose a compact (4X4) sub-image having 4 gray levels is considered and its corresponding Gray Level is defined as in Fig. 3.

2	1	3	4
3	1	4	4
3	2	2	2
1	4	1	4

Fig. 3: A Sample 4 × 4 Subimage

The Gray Level Run Length Matrix $P(i, j|\ \theta = 0^o)$ for the given sub-image is defined in Table 1.

TABLE 1: GRAY LEVEL RUN LENGTH MATRIX $P(I, J|\ \Theta = 0^O)$ FOR THE ABOVE SUB IMAGE

Gray Level Intensity (i)	Run length (j)			
	1	2	3	4
1	4	0	0	0
2	1	0	1	0
3	3	0	0	0
4	3	1	0	0

The reduction in the number of gray levels in a gray image can be done through re-quantization before the accumulation of the Gray Level Run Length Matrix. Before

the run length matrices are accumulated, the gray levels have been quantized into 16 levels in the paper. Fewer levels views the image on a coarse scale, alternatively more levels produce an image with more details. Therefore the performance of a given GLRLM based features, along with the feature ranking, also depends on the number of gray levels used [10]. The basic GLRLM technique is then modified by computing those matrices for four different orientations (θ = 0, π/4, π /2 and 3π /4 radians) and averaging these matrices to produce improved features. These orientations can be obtained by rotating the image in the respective angles and then computing the GLRL matrices. A number of statistical texture features can be computed from the Gray Level Run Length Matrices out of which two important features are considered for the purpose of feature extraction.

1. Short Runs Emphasis:

$$SRE = \frac{\sum_{i=1}^{G}\sum_{j=1}^{R}\frac{p(i,j\mid\theta)}{j^2}}{\sum_{i=1}^{G}\sum_{j=1}^{R}p(i,j\mid\theta)} \quad (2)$$

2. Long Runs Emphasis:

$$LRE = \frac{\sum_{i=1}^{G}\sum_{j=1}^{R}j^2 p(\mathrm{i},\mathrm{j}\mid\theta)}{\sum_{i=1}^{G}\sum_{j=1}^{R}p(i,j\mid\theta)} \quad (3)$$

The symbols denote the following parameters as enlisted below:

1. *P(i, j| θ)* is the *(i,j)*th element of the run length matrix for a direction *θ*
2. *G* is the number of gray levels
3. *R* is the longest run
4. *n* is the number of pixels in the image.

C. *Feature Extraction using Combination of LBP and GLRLM*

The next step of feature extraction involved the combination of LBP technique followed by GLRLM technique. The input images were first fed to the LBP algorithm. The resulting output images are stored and fed as input to the GLRLM algorithm. Finally the output GLRL matrices of the GLRLM algorithm was used to compute the GLRLM texture features. These feature vectors were then stored and processed for the next step of classification. This combinational feature extraction method results in better texture features and hence better classification rate.

D. *Classification using k-Nearest Neighbor Classifier*

A popular non-parametric method which is used for classification and regression analysis is the *k*-Nearest Neighbors (*k*-NN) algorithm. The *k*-NN algorithm measures the distance between an objective point and a set of points in the data set and assigns the objective (test) point to the class that is most common between its *k* nearest neighbors around it. Discussion related to neighbors implies that there must be a distance or dissimilarity measurement that can be computed between samples using the independent variables. The instances involved in the paper consider the most accepted measure of distance i.e. Euclidean distance. The Euclidean distance between any two points *x* and *y* is:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (4)$$

The following figure Fig. 3 shows *k*-NN classification for two different classes A and B. The inner circle encloses three nearest neighbor corresponding to the centre or the test point. Here the majority of the neighbors belong to Class B. Thus the query/test point is said to belong to Class A. The outer circle encloses six nearest neighbors as it corresponds to *k* = 6. But here majority of the neighbors belong to Class A. Hence the test point is classified as Class A for *k* = 6.

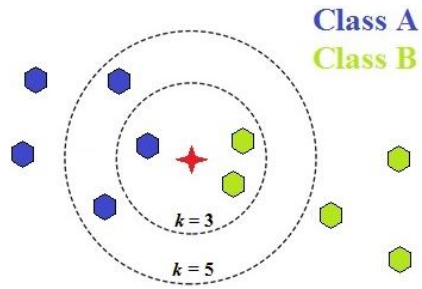

Fig. 4: *k*-Nearest Neighbors Classification for k = 3 and k = 6 using Two Different Classes

If we take the case where k = 1, we need to find the sample in the training dataset that is closest to the nearest neighboring sample. This case of *k* = 1 also corresponds to Euclidean classification. One of the major advantages of using a single nearest neighbor technique to classify samples is the efficiency of the algorithm when the training set consists of a large number of samples [11]. When we have a large volume of data and use an arbitrary classification algorithm, the misclassification error can be reduced to maximum half of that of the traditional 1-NN rule. The optimal value of k primarily depends upon the data. When the value of k is large, the effect of noise on classification is reduced but the boundaries between classes tend to be less distinct [12]. In binary classification scenarios, it is always better to choose k to be an odd number since this avoids tie between the votes. The value of *k* taken for *k*-NN classification in this paper is 3.

E. *Classification using Support Vector Machine (SVM) Classifier*

Support Vector Machines (SVM) are supervised learning models developed initially by Vladimir Vapnik in 1995 [13]. It is used in machine learning with appropriate learning algorithms that analyze data used in classification as well as regression analysis. Basically, a support vector machine creates a separating hyperplane or a family of hyperplanes in a higher or infinite-dimensional space. Thus, an efficient separation is obtained when a particular hyperplane has the largest distance from the nearest training sample of any class. Taking two classes, if a set of training samples are given, each specified for belonging to any one of the two classes, the SVM training algorithm devises a model which appends new samples (test data) to one of the two categories. Another major advantage of using SVM is that it can effectively perform non-linear classifications by using appropriate kernel function. Such a technique implicitly maps the inputs into a higher dimensional feature space. Thus it shows better classification accuracy than k-NN classifier for higher number of membership classes.

The original maximum-margin hyperplane algorithm creates a linear classifier. However, nonlinear classifiers developed by Bernhard Boser, Isabelle Guyon and Vladimir Vapnik used the kernel trick to obtain the separating hyperplanes [14]. Here the algorithm uses a nonlinear kernel function instead of every dot product. The Gaussian radial basis function (rbf) discussed in [15], is used as the kernel function in this paper. The radial basis function on two samples x_i and x_j is of the form:

$$K(\mathrm{x}_i, \mathrm{x}_j) = \exp\left(-\frac{\left\| x_i - x_j \right\|^2}{2\sigma^2} \right) \tag{5}$$

where σ is a free parameter.

Usually the fundamental SVM can be directly applied for two-class tasks. But there is multiple number of texture classes involved in the paper for which the traditional SVM classifiers fails. Hence this classifier has to be extended for classification of multiple classes, alternatively called as Multiclass Support Vector Machine (SVM). The aim of Multiclass SVM is to assign class labels to instances during the training period, where the class labels are brought from a finite set of known elements. The one-versus-all classification was performed where binary classifiers are built to differentiate between one of the previously specified labels and the rest using the logic of "winner takes it all". For instance the first class label values are made equal to 1 while keeping the rest of the class labels as 0. The classification using binary SVM is carried out and the results of the first set of test samples are stored. Similarly the next step makes the set of second class labels as 1 making the other labels are 0 and then binary SVM is carried out. This process is carried forward until all the test features are classified.

III. Results

The programs for the above algorithms were run on MATLAB R2014a on an Intel Core 2 Duo processor and Windows 8 operating system. The texture features were extracted by taking 10 different texture images from the Brodatz database shown in Fig. 5.

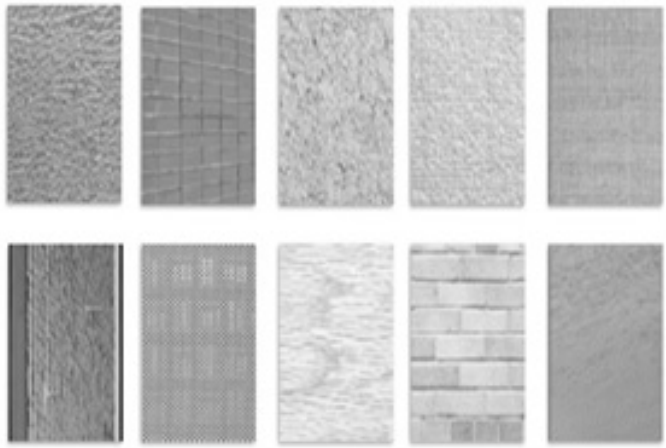

Fig. 5: Texture Images from Brodatz Database Used in the Classification Algorithms

The feature vectors obtained consisted of 720 features in the training as well as testing datasets. The LBP feature vectors consisted of the histogram features. Similarly the GLRL features vector consisted of the SRE and LRE features. For k-NN classification, the value of k was taken to be 3 and it gives the best result from a set of values. Similarly the values of C and gamma (γ) for RBF kernel SVM was set at the default value of 1 for both. The bias (*b*) parameter was set at-3.277. The features are then classified as per the specified classifiers and the classification accuracy is found out by the parameter Classification rate (CR) given by equation 6.

$$CR = \frac{N - P}{N} \times 100 \tag{6}$$

where N = total no. of subimages

P = no. of incorrectly classified subimages

The comparison of classification rate for various feature extraction techniques using different classification algorithms have been show in Table 2.

TABLE 2: CLASSIFICATION PERFORMANCE OF DIFFERENT TEXTURE OPERATORS WITH DIFFERENT CLASSIFIERS

Texture Operator	Clasification Rate (in%)			Computation time (in sec)
	Euclidean	*k*-NN	SVM	
LBP	70.56	72.5	90.41	46.121
GLRLM	13.34	15	23.21	120.766
LBP + GLRLM	89.88	91.25	97.78	170.808

Comparing the above results clearly shows that combination of LBP and GLRM produces much better classification rate as compared to the individual feature extraction techniques. Moreover it is also gives another important observation that non-linear SVM classification provides better classification rate than the Nearest neighbor classification algorithms. The only drawback for the proposed method is the time complexity involved in the feature extraction process (show in Table 2), since it involves two feature extraction stages in it.

Another significant set of observations was obtained by taking the number of training and testing features in different ratios. It can be noted from Table 3 that more is the number of features in the training dataset better is the classification rate. The maximum classification rate was hence obtained by taking 70% features in training dataset and 30% features in testing dataset. This gives rise to the fact that classification accuracy can be improved by developing the complexity of the algorithms as well as by varying the number of data points in the datasets. More number of data in the training set would have enhance classification accuracy but the distinctness of boundaries will be severly affected.

TABLE 3: CLASSIFICATION RATE FOR DIFFERENT RATIO OF TRAINING/TESTING DATASETS

Classifier	**Texture Operator**	**Classification Rate for Different Ratio of Training/Test Data (%)**		
		50/50	60/40	70/30
Euclidean	*LBP*	70.56	71.32	72.46
	GLRLM	13.34	14.55	15.08
	LBP+GLRLM	89.88	90.12	90.99
k-NN	*LBP*	72.50	73.60	74.87
	GLRLM	15.00	16.00	17.18
	LBP+GLRLM	91.25	92.44	93.93
SVM	*LBP*	90.41	90.66	91.57
	GLRLM	23.21	25.34	27.76
	LBP+GLRLM	97.78	98.05	98.41

IV. CONCLUSION

The proposed method combines the texture features of several orientations of the GLRLM along with LBP texture operator resulting in better classification rate. This can be attributed to the generation of improved feature vectors. The best classification rate of 98.41% is observed for the combination of LBP and GLRLM features using SVM, which also shows the efficiency of a non-linear classifier than linear classifiers. An excellent application of such texture classification techniques lies in the field of medical imaging, remote sensing as well as pattern and face recognition. The above algorithms can also be applied to other texture images and databases. The implementation of such algorithms in the wavelet domain has immense scope for improvement in texture classification applications.

REFERENCES

[1] T. Ojala, M. Pietikainen, and D. Harwood, "A comparative study of texture measures with classification based on feature distributions", Symposium on Pattern Recognition, vol. 29 Issue 1, pp. 51-59, January 1996.

[2] M. Varma. and A. Zisserman, "Classifying images of materials: Achieving viewpoint and illumination independence", European Conference on Computer Vision, vol. 02, page III: 255 ff., 2002.

[3] S. Dash, K. Chiranjeevi, U. R. Jena, and A. Trinadh, "Comparative study of image texture classification techniques", IEEE International Conference on Electrical, Electronics, Signals, Communication and Optimization (EESCO), pp. 1-6, January 2015.

[4] Jisy Raju, C. A. D. Durai, "A survey on texture classification techniques", IEEE International Conference on Information Communication and Embedded Systems (ICICES)", pp. 180-184, February 2013.

[5] D. S. Guru, Y. H. Sharath, S. Manjunath, "Texture Features and KNN in Classification of Flower Images", IJCA Special Issue on Recent Trends in Image Processing and Pattern Recognition (RTIPPR), pp. 21-29, 2010.

[6] L. Shutao, J. T. Kwok, H. Zhua, and Y. Wang, "Texture classification using the support vector machines", Pattern Recognition, vol. 36, pp. 2883-2893, 2003.

[7] T. Ojala, M. Pietikainen, and D. Harwood, "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions, in Pattern Recognition", Proceedings of the 12th IAPR International Conference on Computer Vision & Image Processing, IEEE, vol. 1, pp. 582–585,1994.

[8] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns", IEEE Transactions on Pattern Anal. Mach. Intell. 24, pp.971–987, 2002.

[9] M. M. Galloway, "Texture analysis using gray level run lengths," Computer Graphics and Image Processing, vol. 4, pp. 172–179, June 1975.

[10] F. Albregtsen, "Statistical Texture Measures computed from Gray Level Run Length Matrices", Image Processing Laboratory, Department of Informatics, University of Oslo, Nov.1995.

[11] L. Zhou, L. Wang, X. Ge, amd Q. Shi, "A Clustering-Based KNN Improved Algorithm CLKNN for Text Classification", 2nd International Asia Conference on Informatics in Control, Automation and Robotics, IEEE, vol. 3, pp. 212-215, March 2010.

[12] B. S. Everitt, S. Landau, M. Leese, and D. Stahl. (2011, January). Miscellaneous Clustering Methods, in Cluster Analysis, (5th Edition) [Online]. Available: http://onlinelibrary.wiley.com

[13] V. Vapnik, C. Cortes, "Support-vector networks", Machine Learning, vol. 20, Issue. 3, pp. 273–297, 1995.

[14] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers", Proceedings of the fifth annual workshop on Computational learning theory–COLT '92, Association for Computing Machinery (ACM), pp. 144-152, 1992.

[15] J.A.K. Suykens, J. Vandewalle, "Least Squares Support Vector Machine Classifiers", Neural Processing Letters, vol. 9, Issue 3, pp. 293-300, June 1999.