IJICTR International Journal of Information & Communication Technology Research ITRC

Volume 7- Number 3- Summer 2015 (21-30)

Fast GLCM and Gabor Filters for Texture Classification of Very High Resolution Remote Sensing Images

Fardin Mirzapour
Faculty of Electrical and Computer Engineering
Tarbiat Modares University
Tehran, Iran
f.mirzapour@modares.ac.ir

Hassan Ghassemian
Faculty of Electrical and Computer Engineering
Tarbiat Modares University
Tehran, Iran
ghassemi@modares.ac.ir

Received: February 11 , 2014-Accepted: June 21, 2015

Abstract—**In the present research we have used gray level co-occurrence matrices (GLCM) and Gabor filters to extract texture features in order to classify satellite images. The main drawback of GLCM algorithm is its time-consuming nature. In this work, we proposed a fast GLCM algorithm to overcome the mentioned weakness of the traditional GLCM. The fast GLCM is capable of extracting approximately the same features as the traditional GLCM does, but in much less time (about 200 times faster). The other weakness of the traditional GLCM is its lower accuracy in the regions near the class borders. Since features extracted using Gabor filters are more accurate in boundary regions, we combined Gabor features with GLCM features. In this way we could compensate the latter mentioned weakness of GLCM. Experimental results show good capabilities of the proposed fast GLCM and the feature fusion method in classification of very high resolution remote sensing images.**

Keywords- fast GLCM; Gabor filters; texture feature; classification; Remote Sensing.

I. INTRODUCTION

Features commonly used in classification of remote sensing images are categorized into two main groups: spectral features and spatial features. A spectral feature vector of a pixel is a vector whose elements are the reflected energy, from a point in the scene corresponding to the pixel recorded in different spectral bands, or linear/nonlinear combinations of these reflectance values [1]. So, a spectral feature vector is defined only for colored, multispectral (MS), and hyperspectral (HS) images. On the other side, spatial features of a pixel are the ones which are obtained from processing the gray level values of a pixel and its neighbors in a single-band image [2-7]. Thus, this kind of features can be defined for single-band images, such as panchromatic satellite images, as well as individual bands of colored, MS, or HS images. Spatial features used in image processing can be divided into two main categories: texture and shape features. Texture features act as a measure of coarseness, size, and directionality of image details, while the latter assesses the shape of these details [8-10]. However, there is not a clear distinction between these two categories, e.g. features extracted from gray level co-occurrence matrices (GLCM) are known as texture features while they could be used as shape measures too [9, 10].

Texture analysis and classification is one of the active areas in machine vision and image processing, and is used in various applications such as object recognition and tracking [2-4], image retrieval [5, 6],

and satellite image classification [7, 11, 12]. A wide variety of techniques for image texture analysis have been proposed. Keen readers may find good information in [8, 10, 13].

In the present research, we try to utilize two kinds of spatial features in order to classify single-band images: statistical features extracted from GLCM matrices, and structural features obtained using Gabor filters.

GLCM matrices capture image properties related to the second-order statistics of the pixel intensities [14] in an image, and are one of the most well-known texture feature extraction approaches. Despite their popularity and the ability to extract texture context, GLCM features have two main drawbacks; being highly time-consuming, and having relatively low accuracy in the regions near the class borders. To address these deficiencies, we have proposed two solutions: 1) a fast algorithm to extract GLCM features while preserving their quality, and 2) fusing GLCM features with the features obtained from another approach which is more accurate in border regions, i.e. Gabor features [7, 15-20]. These features are obtained through processing the input image in the joint spatial-Fourier domain by applying Gabor filters; without concerning Heisenberg uncertainty inequality which is a known issue when using Fourier transform as a local structural feature descriptor [13]. Note that this paper is an extended version of the work published in [21]. We extended our previous work by providing more detailed discussions, a comparison between complex and real Gabor features, some performance evaluations using analytical computational complexity assessments, and implementing the method on non-mosaic remote sensing panchromatic dataset with natural borders between different land covers.

The outline of the remainder of this paper is as follows. In section II, we briefly introduce GLCM matrices and Gabor filters, and propose a fast algorithm for GLCM calculations. In section III, GLCM and Gabor features are fused to make it possible to use their advantages simultaneously. In addition, the computational complexities of the feature extraction methods are analytically compared. Finally, section IV concludes this work.

II. FEATURE EXTRACTION ALGORITHMS

A. GLCM

One of the simplest statistics of a two dimensional image is the information obtained from its one-dimensional histogram, i.e. the probability of gray level occurrences. One-dimensional histogram does not consider the relationship between pixels exactly, thus it is not a good texture measure. To overcome this weakness, two-dimensional histogram was introduced [14] which is in fact the probability of occurrence of two different gray levels in the neighborhood of a pixel under examination. In this approach, the relationship between pixels is considered more accurately, but it is very time consuming. Again to solve this new problem, an approach was proposed in which between-pixel relationships were considered only in a few predefined directions and distances. To be more accurate, for a pixel with (x_c,y_c) coordination placed at the center of its neighborhood window $\mathbf{W}^{(x_c,y_c)}$, the (i, j)-element of its $\mathbf{GLCM}_{d,\theta}^{(x_c,y_c)}$ matrix is defined as the number of the occurrence of pixels with the gray levels of j, at the distance d, and at the direction θ of pixels with the gray level of i. All these pixels are located in the neighborhood window $\mathbf{W}^{(x_c,y_c)}$. This could be described as

$$\mathbf{GLCM}_{d,\theta}^{(x_c,y_c)}(i,j)=\sum_{\substack{(x_1,y_1)\\(x_2,y_2)}}\delta\left[\mathbf{I}(x_1,y_1),\,i\right].\delta\left[\mathbf{I}(x_2,y_2),\,j\right]$$
$$\text{s.t.}\ \begin{cases}(x_2,y_2)=(x_1,y_1)+(d\cos\theta,d\sin\theta)\\(x_1,y_1)\ \text{and}\ (x_2,y_2)\in\mathbf{W}_{(x_c,y_c)}\end{cases}\qquad(1)$$

in which, δ[.,.] is the Kronecker delta function, $\mathbf{I}$ is a gray level image, i and j are two gray levels from the range $\{1,\ldots,G\}$, $\mathbf{W}^{(x_c,y_c)}$ is the neighborhood window centered at (x_c,y_c), and (x_2,y_2) is a location at the angular distance (d,θ) from (x_1,y_1).

Using (1), for every distance-direction couple, (d,θ), a $G\times G$ $\mathbf{GLCM}_{d,\theta}^{(x_c,y_c)}$ matrix is obtained, in which G is the number of gray levels in the image. After obtaining GLCM matrices for each pixel in the input image, some statistics such as mean, standard deviation, and entropy is extracted from these matrices and are dedicated to the owner pixel. An important parameter in GLCM computations that should be considered is the size of the neighborhood window. Small window sizes, theoretically, result in better discrimination in regions near borders, while in practice they will generate sparse GLCM matrices which cause inaccurate feature extraction process. On the other hand, although large window sizes will result in more accurate extracted features, they cause different classes to mix up in the regions near the class borders. So, choosing appropriate neighborhood window size is an important step in GLCM process.

GLCM features provide good description of image texture, but they need strong processing resources. Many approaches have been proposed to face this problem. The simplest one is to reduce G by re-quantizing the gray levels of the image pixels. This causes the dimensions of GLCM matrices to decrease. In addition, this preprocessing phase will reduce the sparsity of GLCM matrices. Another approach is to consider less distance-direction couples, (d,θ). It is shown that the features extracted from GLCM matrices corresponding to "(d,θ)=(1,0)" are enough to describe texture of most images [14].

Although GLCM were introduced about four decades ago, it is a strong method to extract texture features and still many attempts are being made to improve its speed and performance, or to use it in combination with newer methods. Here, we will propose a new way to conquer the computational resource consuming nature of GLCM while maintaining the strength of the extracted features. In other words, we make it faster while the accuracy of the image classification using these features (as a criterion to assess the quality of the extracted features) is not affected.

The proposed method is based on the concept that the features of spatially-close pixels are closely related. The simplest type of dependency is linear dependency. With assumption of this type of correlation, we can calculate GLCM matrices (and then extract the related features from them) only for a few pixels (let's call them key pixels) by skipping pixels with a step size of L_s in row and column. Then the extracted features for these key pixels are assigned to all the pixels in their neighborhood with a pyramidal weight matrix such as the matrix shown in Fig 1.(*a*). Finally, the feature vector associated to each non-key pixel *p* is the weighted sum of all feature vectors from all the weighting windows which embrace *p*. In other words, we calculate GLCM features for the key pixels and use interpolation technique to estimate the GLCM features of all other pixels. This could be shown as

$$\mathbf{f}(p) = \sum_{i=1}^{n} a_i(p)\,\mathbf{f}(x_i) \qquad (2)$$

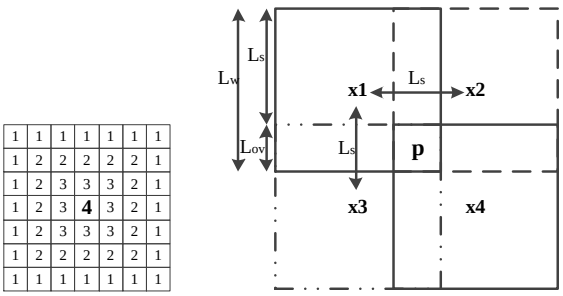
in which, $\mathbf{f}(p)$ is the feature vector of any non-key pixel *p*; x_i is the key pixel located at the center of each of the overlapping weighting windows which include *p*; and $a_i(p)$ is the weight associated to pixel *p* in the weighting window centered at x_i (Fig 1). For each non-key pixel, the number of the overlapping windows, *n*, is always less than or equal to 4; e.g. for the sample point *p* shown in Fig 1.(*b*), *n*=4. The last point that should be mentioned here is that, GLCM features for the key pixels are calculated in the original image not in a subsampled image. However, as said before, the neighboring key pixels are separated by L_s rows and/or columns. Also, we select the pixel located at $(x,y)=(\lfloor L_s/2 \rfloor, \lfloor L_s/2 \rfloor)$ as the starting key pixel ($\lfloor . \rfloor$ denotes the integer part operator).

In order to evaluate the proposed fast GLCM, we implemented the algorithm on three single-band images with different sizes: (*a*) a 1024×1024-pixel image, (*b*) a 512×512-pixel image, and (*c*) a 256×256-pixel image (Fig 2). Each of these images contains 5 different textures from Brodatz set. The parameters used in this implementation are selected as follows (see TABLE I):

- The size of the GLCM extraction window, $\mathbf{W}^{(x_c,y_c)}$, is considered to be 33×33. Although more complex methods for window size selection, such as adaptive algorithms [22], could be adopted, we used a fixed size window for simplicity.
- As mentioned before, to reduce the computational burden of GLCM matrices extraction process, and also to decrease the sparsity of GLCM matrices, input image gray levels are usually re-quantized in order to reduce *G*. Here, *G* is reduced from 256 levels to 32 levels. So the generated GLCM matrices will be 32×32.
- Contrast, Correlation, Energy, Homogeneity, Entropy, and Variance, are the 6 features extracted from GLCM matrices. According to the authors' experiences, it is good practice to apply a principal component transform (PCT) on these features to reduce the redundancy.

To evaluate the effect of the main parameter of the fast GLCM algorithm, i.e. the skip length (L_s), on the extracted features quality, L_s is picked from the range 1 to 16. Note that, "L_s=1" corresponds to normal GLCM. To assess the quality of the extracted features, we have used overall ML classification accuracy as quality measure. In order to train ML classifier, 5% of pixels are selected randomly. The other 95% of pixels are used as test samples. The overall ML classification accuracy and the relative GLCM feature extraction times are illustrated in Fig 3 against various skip lengths, L_s=1,…,16. The given processing times are normalized to the processing time for the case of L_s=1, i.e. normal GLCM. As can be seen, fast GLCM algorithm can significantly reduce the processing time (approximately, by a factor of L_s^2) while preserving the features quality.

A point that should be mentioned here is the effect of the minimum size of objects (connected areas of the same texture) in the input image, on the maximum value of the algorithm parameter, L_s. It goes without saying that the skip length should be less than the dimensions of the smallest connected area of the same texture in the image; otherwise the smaller regions may be dismissed. Thus a prior knowledge about the image is required to select an appropriate value for L_s.



(*a*) (*b*)

Fig 1. (*a*) Linear (pyramidal) weighting window for a window length of L_w=7, and (*b*) four $L_w \times L_w$ weighting windows with an overlap of length L_{ov}; features of pixel p are the weighted sum of that of x_is.

TABLE I. PARAMETERS USED IN EVALUATING FAST GLCM ALGORITHM

Parameter	value
(d,θ) in (1)	(1,0)
G (The number of gray levels of the image after re-quantization)	2^5 bits = 32 levels GLCM matrices: 32×32
Features extracted from GLCM matrices	Primary features (6 features): Contrast, Correlation, Energy, Homogeneity, Entropy, Variance
	Final features (5 features): First, applying PCT to the primary features and then selecting the first 5 components
GLCM extraction window	33×33
Weighting window	33×33 pyramid (Fig 1.)
L_s (skip length)	1,…,16

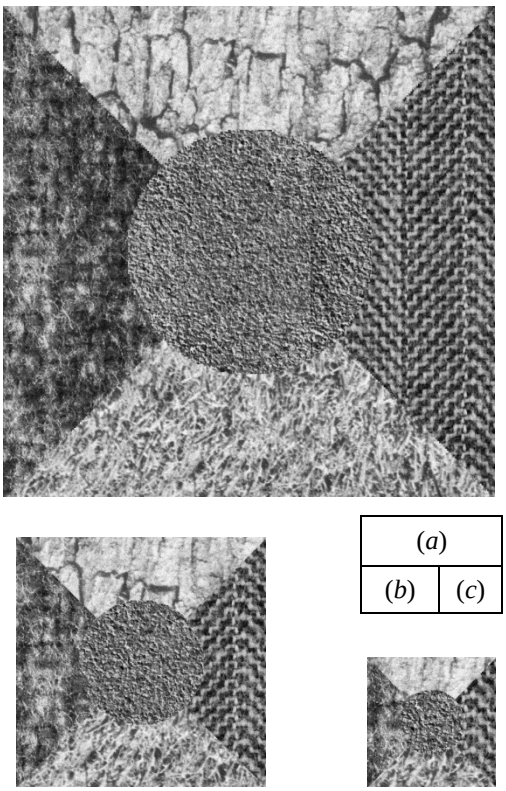


Fig 2. Single-band images, containing 5 different textures synthetized from Brodatz set, used to evaluate the proposed fast GLCM algorithm: (*a*) 1024×1024, (*b*) 512×512, and (*c*) 256×256.

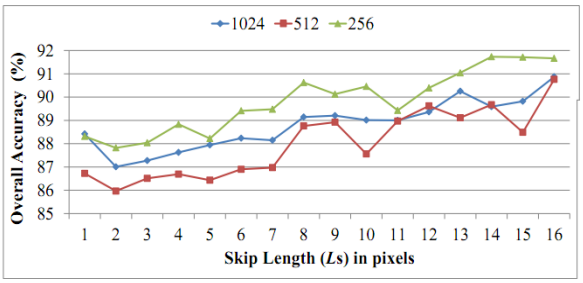
The curves plotted in Fig 3.(*a*) show variations in classification accuracy versus L_s for all test images of Fig 2. These trends are approximately increasing for all 3 cases. As mentioned before, if we take a look at (2), we see that fast GLCM features are actually estimations obtained from linear interpolations of key-pixels' features. Employing linear interpolation always causes smoothness. Therefore, the proposed GLCM features will be smoother than traditional GLCM features. Consequently, the proposed GLCM features are able to provide more homogeneous classification maps with higher classification accuracies.

On the other hand, as Fig 3.(*b*) suggests, increasing skip length will dramatically increase the speed of the algorithm. In fact, the algorithm is faster approximately by a factor of L_s^2. This is because GLCM features are calculated for N/L_s out of N pixels, where N is the number of all pixels in the image. This will be discussed analytically in subsection III.C.

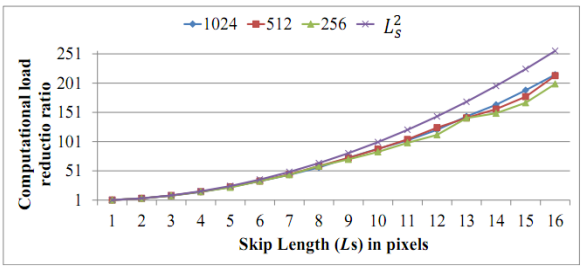
To sum up, according to the obtained results of the experiments and the above discussions, for the test images shown in Fig 2, choosing L_s=16 would be an appropriate choice.

B. Gabor Filters

Gabor filters have been widely used in different areas of image processing such as texture classification, edge detection, fingerprint identification, and image coding [7, 15-20]. Also, different methods have been developed to use Gabor filters in image classification [16, 18]. In [7], Gabor wavelets are utilized to extract image texture features. The idea is based on detecting linear directional elements in the image.



(*a*)



(*b*)

Fig 3. (*a*) Overall ML classification accuracy for the images shown in Fig 2 using fast GLCM features, (*b*) the computational load reduction ratio; The horizontal axis shows the main parameter of the fast GLCM algorithm, i.e. skip length (L_s); The legends represent the dimensions of the input images.

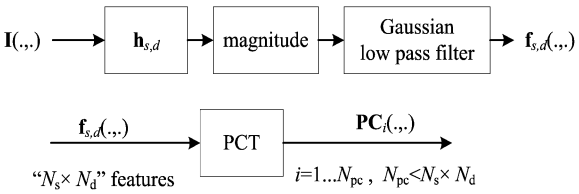


Fig 4. Gabor features extraction process. Refer to context for details.

In this method, a set of wavelets $\{\mathbf{h}_{s,d} | s = 1 \ldots N_s, d = 1 \ldots N_d\}$ is generated using (4) from a mother wavelet given by (3):

$$\varphi(x, y) = \frac{1}{2\pi\sigma_x \sigma_y} \exp\left\{-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right) + 2\pi \mathrm{j} U_h x\right\} \tag{3}$$

$$\mathbf{h}_{s,d}(x, y) = \left(U_h / U_l\right)^{\frac{-s}{s-1}} \cdot \varphi(X, Y) \tag{4}$$

where,

$$\begin{cases} X = \left(\frac{U_h}{U_l}\right)^{\frac{-s}{s-1}} \left[(x - x_0)\cos\left(\frac{d\pi}{N_d}\right) + (y - y_0)\sin\left(\frac{d\pi}{N_d}\right)\right] \\ Y = \left(\frac{U_h}{U_l}\right)^{\frac{-s}{s-1}} \left[-(x - x_0)\sin\left(\frac{d\pi}{N_d}\right) + (y - y_0)\cos\left(\frac{d\pi}{N_d}\right)\right] \end{cases}$$

and, s=1,…,N_s and d=1,…,N_d are the scale and direction parameters of wavelets; (x_0,y_0) is the filter center coordination in the spatial domain; U_l and U_h are respectively the minimum and maximum center

frequency of filters on the horizontal axis in Fourier domain.

Then the whole input image is fed as the input of the wavelet set (see Fig 4). Moreover, in order to reduce the within-class variances, and consequently reducing classification errors, a Gaussian LPF is applied to these values [7]. To reduce the number of features and also to lower the information redundancy –which is due to the overlapping of filters– principal component transform (PCT) is usually applied to the outputs of Gabor wavelets. The main characteristic of PCT is that the output components of the transform are theoretically uncorrelated. Typically, after applying this transform to input feature vectors, a number of the output components (elements of the output vector) which their cumulative sum of energy is bigger than a user defined threshold are preserved and the rest are discarded. The number of preserved components depends on the selected threshold, as well as the input data characteristics.

Since the filters output values are complex, we may use the magnitude of these values or their real parts. According to our experiences, the magnitudes of the complex global Gabor features are much more powerful than the real Gabor features. To show this, we compared these two approaches. The results are depicted in Fig 5. As can be seen, magnitudes of complex Gabor features are far preferable to the real features.

Two key parameters of Gabor filters are the number of scales and directions, N_s and N_d. To find the optimum values for these parameters, we classified the images shown in Fig 2 using Gabor features extracted for different values of N_s and N_d, by applying an ML classifier. The parameters used in this implementation are listed in TABLE II. Again, the overall classification accuracy is used as the selection criteria. According to Fig 6, although the optimal choice of N_s and N_d differs for different input images, there is an obvious distinction between accuracy values for "$N_d \geq 4$ and $N_s \geq 6$". So, the boundary values of $N_d=4$ and $N_s=6$, would be appropriate choices regardless of the input data.

III. IMPLEMENTATION

In the previous section, we saw that by utilizing the proposed fast GLCM algorithm we were able to overcome the main drawback of GLCM –i.e. its slow nature– while benefiting from its strength in extracting texture features. Also, according to the diagrams in Fig 3, a good choice for skip length in fast GLCM was $L_s=16$. In addition, we saw that selecting the number of directions and scales, $N_d=4$ and $N_s=6$, for Gabor filters could be a practical option.

Here, we will extract fast GLCM and Gabor features from two sets of images: a set of images synthesized from different Brodatz textures (see Fig 2), and two panchromatic satellite image gathered over Tehran/Iran (Fig 10). It should be noted that the synthesized images shown in Fig 2.(*b*), and (*c*) are not the resized versions of the image depicted in Fig 2.(*a*). Actually they all include Brodatz textures of the same resolution.

The test setup flowchart is illustrated in Fig 7.

A. *Implementation on Brodatz textures*

Fast GLCM and Gabor features are extracted from the images shown in Fig 2. The parameters are selected as shown in Fig 7. The results are illustrated in Fig 8.(*a*1 and *b*1) through Fig 8.(*a*3 and *b*3), respectively for the images shown in Fig 2.(*a*) through Fig 2.(*c*). As can be seen in these Figs, Gabor features have good ability to find class boundaries, but there is a tendency to generate small speckle like objects in output class maps.

On the other hand, GLCM features are less accurate in areas close to class borders, but small objects in output class maps are rare. Therefore, it seems that by combining these two types of features, we may be able to use the advantages of them both.

TABLE II. PARAMETERS USED IN GABOR FEATURE EXTRACTION

Parameter	Value
Filter parameters	21×21 window, U_l =0.01, U_h=0.049
Number of filters	$N_d \times N_s$ (N_d directions and N_s scales)
Extracted features	Primary features: $N_d \times N_s$ features
	Final features: Extracted by applying PCT to the primary features and using a threshold level of 95% (the number of final features, N_{pc}, depends on this threshold and characteristics of input features)
Gaussian LPF	($w \times w$), $\sigma_x = \sigma_y = w/3$, $w = 21$

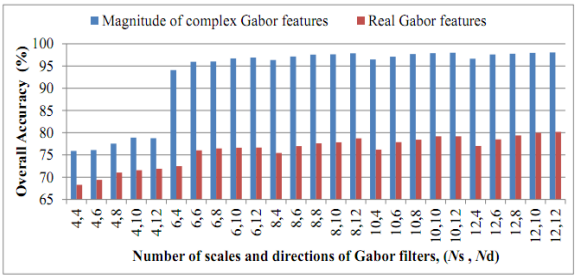


Fig 5. Overall classification accuracies with different numbers of scales (N_s) and directions (N_d) for "magnitudes of complex Gabor features" vs. "real Gabor features".

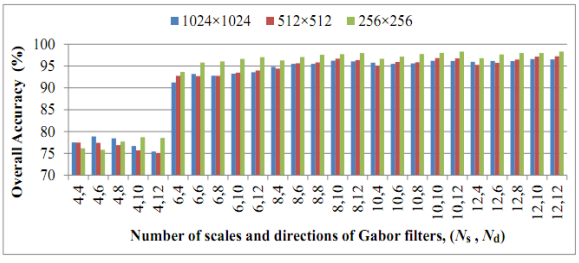


Fig 6. Overall ML classification accuracy for the images shown in Fig 2, using Gabor features for different numbers of scales (N_s) and directions (N_d).

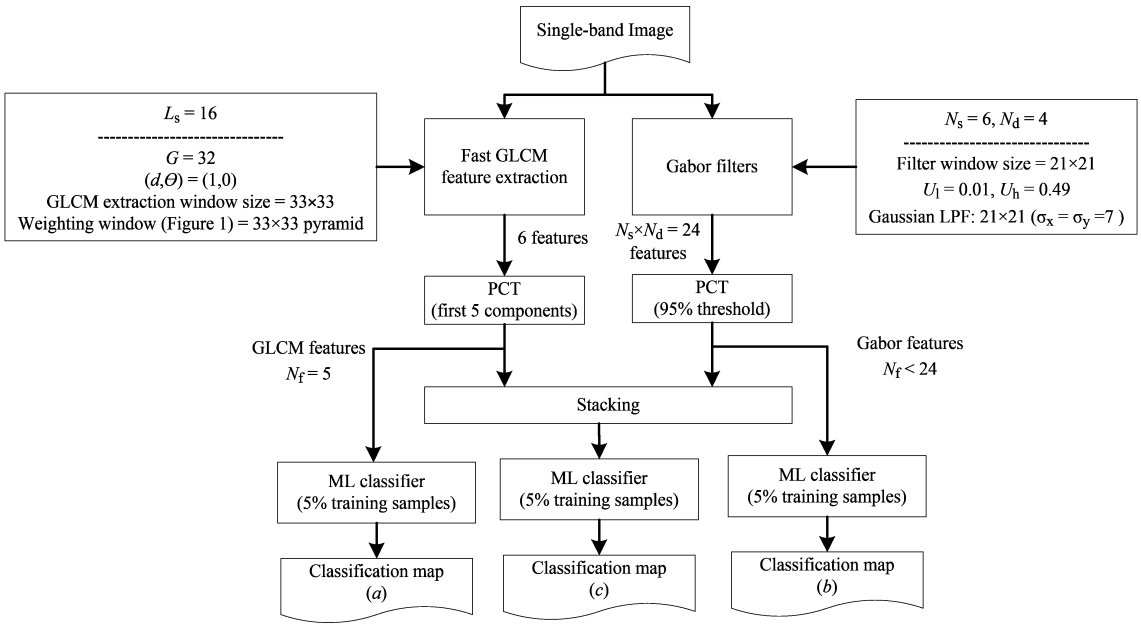


Fig 7. The proposed algorithm with parameters values.

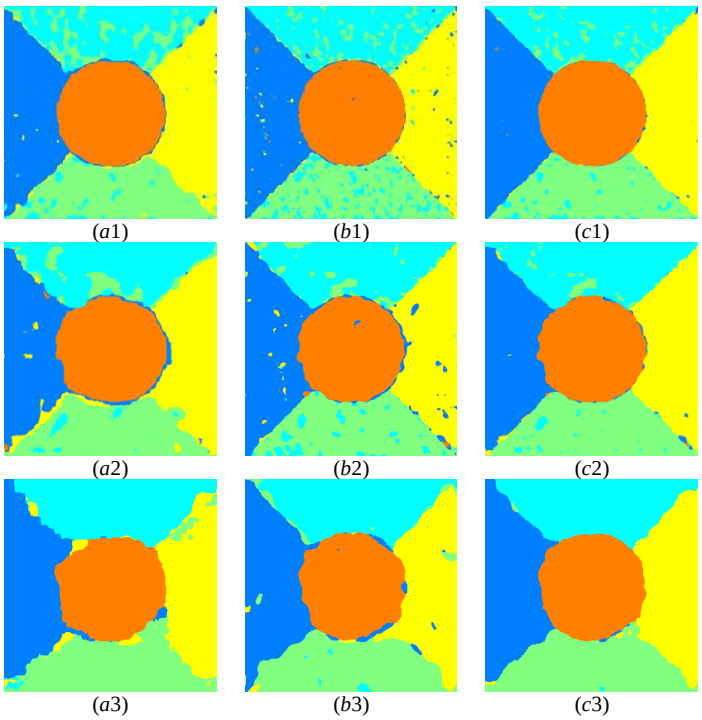


Fig 8. ML classification maps for the images shown in Fig 2, using (*a*) fast GLCM features, (*b*) Gabor features, and (*c*) fused features; Upper row shows results for the 1024×1024-pixel image, middle row for the 512×512-pixel image, and lower row for the 256×256-pixel image.

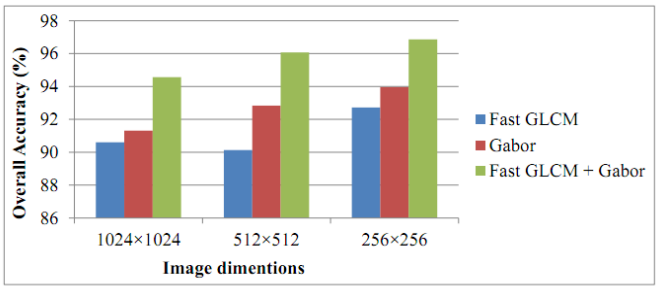


Fig 9. Overall ML classification accuracy for the images shown in Fig 2 using Gabor, Fast GLCM, and fused features (Fast GLCM + Gabor); 5% of pixels for each case are selected randomly to train ML classifier.

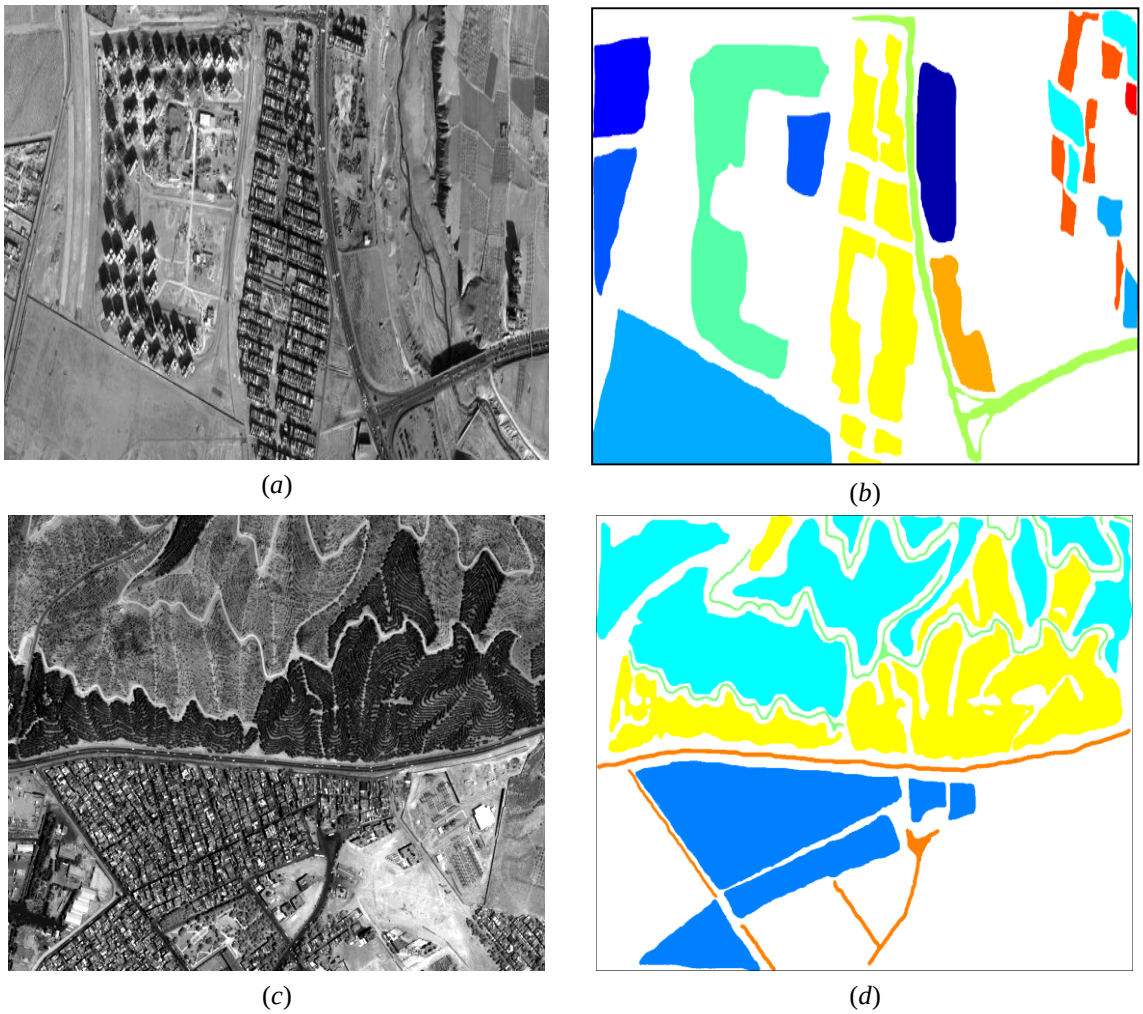


Fig 10. (*a*) PAN satellite image of north-west of Tehran/Iran, a 988×1890-pixel scene containing 11 different land covers; (*b*) the ground truth map (GTM) of (*a*); (*c*) PAN satellite image of north-east of Tehran/Iran, an 822×1154-pixel scene containing 5 different land covers; (*d*) the GTM of (*c*).

To verify this idea, we fused Gabor and GLCM feature vectors by simply stacking them. Then we classified these new vectors. The results are shown in Fig 8.(*c*1) through Fig 8.(*c*3). Also, in Fig 9, overall ML classification accuracies are illustrated for different feature sets. These results clearly confirm the superiority of the fused feature vectors over the individual feature vectors.

B. Implementation on satellite data

The classification results for the synthesized images in the previous subsection demonstrated the power of the extracted features and the fusion idea. Now we will try to implement the proposed fast GLCM feature extraction method and the fusion technique (see Fig 7) on two panchromatic (PAN) satellite images. The scenes are subsets of a large PAN data gathered over Tehran/Iran, with 1-meter spatial resolution. The first scene – Fig 10.(*a*) – corresponds to a region located in north-west of Tehran, and contains 908×1892 pixels and 11 different land cover classes. The ground truth map of the data is shown in Fig 10.(*b*). The second scene – Fig 10.(*c*) – belongs to a region located in north-east of Tehran. This scene has 822×1154 pixels and contains 5 land cover classes. The corresponding ground truth map is depicted in Fig 10.(*d*). The output ML classification maps are depicted in Figs 11 and 12, respectively: (*a*) the classification map using fast GLCM features, (*b*) the map obtained from Gabor features, and (*c*) the map resulted from fused features. The overall classification accuracies are given in TABLE III.

As the table shows, for the first PAN data, the overall accuracy provided by GLCM is much lower than that delivered by Gabor features (78.86% versus 92.81%). Moreover, the classification map of Gabor features is much more satisfactory through visual inspection and the borders' of the classes are much more preserved. However, speckle-like errors are more on its map.

TABLE III. OVERAL ACCURACIES (OA) OF ML CLASSIFICATION USING DIFFERENT FEATURES. FOR TRAINING THE CLASSIFIER, 5% OF THE LABELED SAMPLES ARE RANDOMLY SELECTED.

OA (%)	Features		
	Fast GLCM	Gabor	both
PAN1 (Fig 10.(*a*))	78.86	92.81	96.05
PAN2 (Fig 10.(*c*))	88.37	95.70	98.25

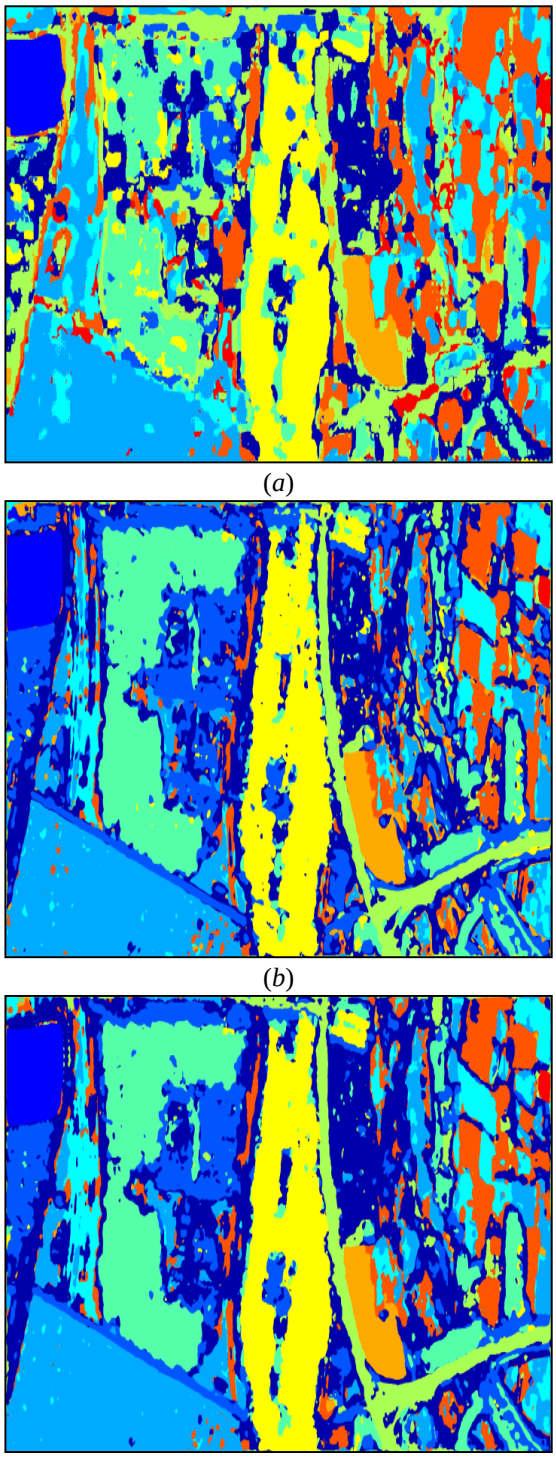


Fig 11. Output ML classification map for the PAN image shown in Fig 10.(*a*) using: (*a*) Fast GLCM, (*b*) Gabor, and (*c*) fused features; 5% of labeled pixels are selected randomly to train the classifier.

The classification map obtained using both types of features – Fig 10.(*c*) – not only has a higher accuracy in terms of OA (96.05%) , but is more homogeneous than the map of Gabor (this is an inheritance from GLCM features) and also has inherited the border preservation property from Gabor features.

The above discussion is also valid for the second PAN data. Therefore, we can conclude that by combining GLCM features with Gabor features extracted from PAN satellite images, we are able to achieve the classification map homogeneity offered by GLCM features while preserving class borders provided by Gabor features.

C. Computational complexity assessment

As shown in Fig 3.(*b*), the proposed fast algorithm for GLCM feature extraction process reduces the computational load approximately by a factor of L_S^2, where L_s is the skip parameter of the algorithm. In this subsection, we will compare the computational load of feature extraction processes, i.e. GLCM, fast GLCM, Gabor, and "fast GLCM+Gabor", analytically.

TABLE IV and TABLE V show the computational complexity of GLCM and Gabor feature extraction processes. As can be seen, the computational complexity of GLCM is of the order of $O(G^2)$, where G is the number of gray levels of the input image after re-quantization process in GLCM (see TABLE I and subsection II.A).

TABLE IV. ORDER OF COMPUTATIONAL COMPLEXITY FOR GLCM ALGORITHM.

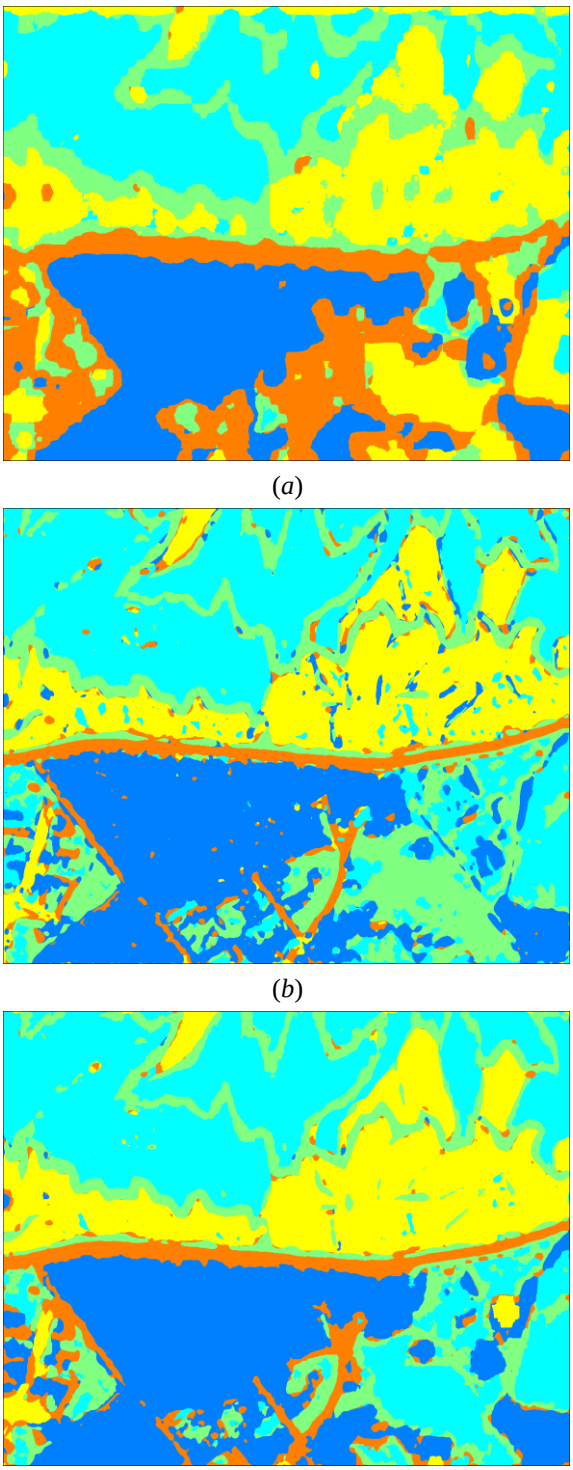
Process	**Number of Operations**		
GLCM matrix generation	$w(w\text{-}1) \approx w^2$		
Feature extraction	**Multiplication**	**Summation**	**Other**
Contrast	$2G^2$	$2G^2$	-
Correlation	$4G^2$	$3G^2$	-
Energy	G^2	G^2	-
Homogeneity	G^2	$3G^2$	-
Entropy	G^2	G^2	G^2
Variance	$2G^2$	$2G^2$	-
All features	$11G^2$	$12G^2$	G^2
TOTAL (for each pixel)	w^2 (Comparisons) $+11G^2$ (Multiplications) $+ 12G^2$ (Summations) $+ G^2$ (log)		
Order of complexity	With the assumption of w having the same order of magnitude as G: $\boldsymbol{O(G^2)}$		

Note: G is the number of the gray levels of the input image after initial re-quantization, and w is the length of the neighborhood window in GLCM feature extraction

TABLE V. ORDER OF COMPUTATIONAL COMPLEXITY FOR GABOR FEATURE EXTRACTION ALGORITHM.

Parameter	**Value**
Number of 2D FFT operations (for the whole image)	$N_xN_y \log_2(N_xN_y)$
Number of Gabor filters	N_sN_d
TOTAL number of operations (for the whole image)	$3N_xN_y \log_2(N_xN_y)\ N_sN_d$
Order of complexity	$\boldsymbol{O(\log_2(N_xN_y))}$

Note: N_x and N_y are the dimensions of the input image, and N_s and N_d are the number of scales and directions of Gabor wavelets, respectively



(*a*)

(*b*)

(*c*)

Fig 12. Output ML classification map for the PAN image shown in Fig 10.(*c*) using: (*a*) Fast GLCM, (*b*) Gabor, and (*c*) fused features; 5% of labeled pixels are selected randomly to train the classifier.

Similarly, the computational complexity for Gabor feature extraction process is of the order of $O(\log_2(N_x N_y))$, in which N_x and N_y are the dimensions of the input image. The calculation of the computational complexity for fast GLCM is straight forward: $O(G^2/L_S^2)$. This is approximately the same result as that of Fig 3.(*b*). TABLE VI summarizes the discussion and as can be seen, the proposed method (Gabor + Fast GLCM) has good performance in terms of computational load, too.

TABLE VI. ORDER OF COMPUTATIONAL COMPLEXITY FOR DIFFERENT METHODS OF FEATURE EXTRACTION.

Method	Order of complexity	
	Parametric	Numerical example for our experiments[(1)]
GLCM	$O(G^2)$	1024
Fast GLCM	$O(G^2/L_s^2)$	4
Gabor	$O(\log_2(N_x N_y))$	20
Gabor + Fast GLCM	$O(\log_2(N_x N_y)+G^2/L_s^2)$	24

[(1)] G=32, L_s=16, N_x=N_y=1024.

Note: G and w are respectively the number of the gray levels of the input image after initial re-quantization and the length of the neighborhood window in GLCM feature extraction. N_x and N_y are the dimensions of the input image, and N_s and N_d are the number of scales and directions of Gabor wavelets, respectively.

IV. CONCLUSIONS

In this paper, we tried to utilize two well-known methods for extracting texture features from single-band satellite images: GLCM and Gabor filters. Although the traditional GLCM method has good performance in texture feature extraction, it is very time consuming. Here, we proposed a fast GLCM algorithm which significantly improved the speed of GLCM: about 200 times faster (corresponding to the skip length of L_s=16). This increase in speed was obtained while preserving the quality of extracted features.

The overall ML classification using the extracted features was used as the measure of quality for the features. The classification results showed that Gabor features are more powerful than GLCM features in the areas close to the class borders, while GLCM features are preferable in the areas within classes. Using these findings, we could test the idea of fusing these two types of features in order to benefit the advantages of both. The implementation results were acceptable and confirmed the idea. In addition, we compared the computational complexity of the feature extraction methods and showed that the proposed method has a very good performance in terms of computational load, too.

REFERENCES

[1] D. A. Landgrebe, *Signal theory methods in multispectral remote sensing*: John Wiley & Sons, 2003.

[2] X. Jianxiong, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "SUN database: Large-scale scene recognition from abbey to zoo," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010, pp. 3485-3492.

[3] Y. Lee, D. K. Han, and H. Ko, "Reinforced AdaBoost Learning for Object Detection with Local Pattern Representations," *The Scientific World Journal*, vol. 2013, 2013.

[4] V. Takala and M. Pietikainen, "Multi-object tracking using color, texture and motion," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, 2007, pp. 1-7.

[5] F. Qiao, C. Wang, X. Zhang, and H. Wang, "Large Scale Near-Duplicate Celebrity Web Images Retrieval Using Visual and Textual Features," *The Scientific World Journal*, vol. 2013, 2013.

[6] S. Murala, R. P. Maheshwari, and R. Balasubramanian, "Local Tetra Patterns: A New Feature Descriptor for Content-Based Image Retrieval," *Image Processing, IEEE Transactions on*, vol. 21, pp. 2874-2886, 2012.

[7] F. Mirzapour, and H. Ghassemian, "Texture Feature Extraction in Satellite Images using Gabor Wavelets," in *Machine Vision and Image Processing Applications* (MVIP 2003), *The 2nd Conference On*, Tehran, Iran, 2003 (PRINTED IN PERSIAN).

[8] J. Zhang and T. Tan, "Brief review of invariant texture analysis methods," *Pattern Recognition*, vol. 35, pp. 735-747, 3// 2002.

[9] A. Materka and M. Strzelecki, "Texture analysis methods–a review," *Technical university of lodz, institute of electronics, COST B11 report, Brussels*, pp. 9-11, 1998.

[10] M. Yang, K. Kpalma, and J. Ronsin, "A survey of shape feature extraction techniques," *Pattern recognition*, pp. 43-90, 2008.

[11] X. Hu, C. V. Tao, and B. Prenzel, "Automatic segmentation of high-resolution satellite imagery by integrating texture, intensity, and color features," *Photogrammetric engineering and remote sensing*, vol. 71, p. 1399, 2005.

[12] M. C. Alonso, M. A. Sanz, and J. A. Malpica, "Classification of high resolution satellite images using texture from the panchromatic band," in *Advances in Visual Computing*, ed: Springer, 2007, pp. 499-508.

[13] M. Tuceryan and A. K. Jain, "Texture analysis," *Handbook of pattern recognition and computer vision*, vol. 2, pp. 207-248, 1993.

[14] R. M. Haralick, K. Shanmugam, and I. H. Dinstein, "Textural Features for Image Classification," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. SMC-3, pp. 610-621, 1973.

[15] T. P. Weldon, W. E. Higgins, and D. F. Dunn, "Efficient Gabor filter design for texture segmentation," *Pattern Recognition*, vol. 29, pp. 2005-2015, 12// 1996.

[16] T. P. Weldon, W. E. Higgins, and D. F. Dunn, "Gabor filter design for multiple texture segmentation," *Optical Engineering*, vol. 35, pp. 2852-2863, 1996.

[17] D. Dunn, W. E. Higgins, and J. Wakeley, "Texture segmentation using 2-D Gabor elementary functions," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, pp. 130-149, 1994.

[18] S. E. Grigorescu, N. Petkov, and P. Kruizinga, "Comparison of texture features based on Gabor filters," *Image Processing, IEEE Transactions on*, vol. 11, pp. 1160-1167, 2002.

[19] M. Idrissa and M. Acheroy, "Texture classification using Gabor filters," *Pattern Recognition Letters*, vol. 23, pp. 1095-1102, 2002.

[20] V. Risojević, S. Momić, and Z. Babić, "Gabor Descriptors for Aerial Image Classification," in Adaptive *and Natural Computing Algorithms*. vol. 6594, A. Dobnikar, U. Lotrič, and B. Šter, Eds., ed: Springer Berlin Heidelberg, 2011, pp. 51-60.

[21] F. Mirzapour and H. Ghassemian, "Using GLCM and Gabor Filters for Classification of PAN Images," in *Electrical Engineering (ICEE), 21st Iranian Conference on*, 2013, pp.1-6.

[22] J.-L. Lotti and G. Giraudon, "Adaptive window algorithm for aerial image stereo," in *Spatial Information from Digital Photogrammetry and Computer Vision: ISPRS Commission III Symposium*, 1994, pp. 517-524.

Fardin Mirzapour received his B.Sc. in communications systems engineering from Sharif University of Tech., Tehran, Iran, in 1999, and M.Sc. from Tarbiat Modares University, Tehran, Iran, in 2002. After spending the subsequent years working in research centers and teaching undergraduate courses, he started his Ph.D. degree in 2012 at Tarbiat Modares University, and graduated in 2015. His research interests include pattern recognition, remote sensing, and signal and image processing.

Hassan Ghassemian received his B.Sc. degree from Tehran College of Telecommunication in 1980 and the M.Sc. and Ph.D. degree from Purdue University, West Lafayette, USA in 1984 and 1988 respectively. Since 1988, he has been with Faculty of Computer and Electrical Engineering at Tarbiat Modares University in Tehran, Iran, where he is a Professor of Electrical and Computer Engineering. Dr. Ghassemian has published more than 400 technical papers in peer-reviewed journals and conference proceedings. He has trained more than 120 M.Sc. and Ph.D. students who have assumed key positions in software and computer system design applications related to signal and image processing in the past 27 years. His current research interests are Multi-Source Signal/Image Processing, Information Analysis and Remote Sensing.