Texture Segmentation using Ant Tree Clustering

Arshad Hussain Channa[1], Nasir Mahmood Rajpoot[2], Kashif Mahmood Rajpoot[3]
[1]Faculty of Computer Science & Engineering, GIK Institute, PAKISTAN [*ahchanna@yahoo.com*]
[2]Department of Computer Science, University of Warwick, UK [*nasir@dcs.warwick.ac.uk*]
[3]Faculty of Computer Science & Engineering, GIK Institute, PAKISTAN [*kashif.rajpoot@yahoo.com*]

***Abstract*— Motivated by the self-assembling behavior of real ants, we present a novel algorithm for texture segmentation which is based on ant tree clustering of wavelet features. In a pattern recognition setting, wavelet features are extracted using either of the two subband filtering methods: discrete wavelet transform (DWT) or discrete wavelet packet transform (DWPT). The feature classification process is inspired by the self-assembling behavior observed in real ants where ants progressively become attached to an existing support and then successively to other attached ants thus building trees based on the similarity of feature vectors. The results thus obtained compare favorably to those of other recently published filtering based texture segmentation algorithms.**

Key words: texture segmentation & analysis, wavelet transform, feature extraction, ant tree clustering.

I. Introduction

Image segmentation is an important stage of early visual information processing where the task is to separate the given input image into multiple regions often on the basis of similar characteristics of pixels belonging to the same region and/or dissimilar characteristics of pixels belonging to any two different regions. Image texture can be considered as a function of spatial variation in pixel intensities. It is hard to find a universally agreeable definition of texture in the literature because of a long list of its properties. Nevertheless, texture contains important clues about image region description and can be described by properties like uniformity, density, coarseness, roughness, regularity, linearity, directionality, frequency, and phase gathered from the appearance of an image region [1].

In texture analysis, a texture measure is used for separating the image into distinct regions through texture classification and/or texture segmentation. In the latter, unlike the former, the objective is not to identify the different textures but only to segment them. With the increasing popularity of digital libraries, multimedia databases, texture analysis has become a focused area of research because of its utilization in numerous industrial and real-world applications [2]: content-based image retrieval, medical image analysis, industrial inspection, remote sensing, and document processing, to name a few only.

Over the last two decades or so, researchers working in the area of texture analysis have proposed various approaches which are typically composed of two main stages: (*i*) feature extraction and (*ii*) feature classification. In the first stage, characteristic attributes (*features*) representing the texture are gathered through the means of statistical, structural, geometric, model-based, and/or filtering approaches [3]. In the second stage, a supervised or unsupervised classifier is used depending upon the nature of the application. Due to the complexity of texture analysis problem, no single approach of feature extraction and feature classification works for all types of texture and application areas. This is the driving force behind research efforts made in this area.

In this paper, we employ discrete wavelet transform (DWT) as a filtering approach for feature extraction. The choice is made because of its superior results and potential for a diverse kind of textures as compared to many other filtering techniques (Gabor, Fourier, ring/wedge, multi-channel, etc.) [3]. At the second stage of feature classification, the ant tree clustering (ATC) algorithm is developed. The algorithm is inspired by the behavior observed in real-world ants. Such natural systems have been used to solve many problems [4]. For instance, different species have developed social behavior to tackle the problem of gathering objects of individuals, like in brood sorting or cemetery organization in real ants [5], and other problems like traveling salesman problem [6]. This paper shows how to adapt the self-assembling behavior of ants to the segmentation problem where the data should be hierarchically organized in a tree. This model is based on the ability of ants to build live structures with their bodies. Each ant represents a feature vector and is initially placed on a fixed point, which is the root of the tree. Based on its similarity with the ants already attached to the root, the ants move and attach themselves to the tree. This behavior has been demonstrated in [7] and used in [8] to estimate the number of clusters a given data can be partitioned into. In this work, the number of partitions is fixed to the actual number of clusters present in the data. The parameters used to calculate similarity of ants are tuned iteratively to produce optimal partitioning of data. The results produced through the ant clustering algorithm are shown to be comparable to the commonly used *k*-means clustering algorithm. A major disadvantage of the *k*-means algorithm, however, is that it can get stuck in local minima in high-dimensional vector spaces. The proposed algorithm uses similarity scope for partitioning the data set and uses a probabilistic approach to avoid local minima.

A brief overview of wavelets in the context of texture analysis is presented in the next section. Section III describes the core of the ant tree clustering algorithm and gives a brief discussion of the algorithm's complexity. The experimental setup is described in Section IV, while experimental results are presented and analyzed in the following section. The paper ends with concluding remarks and some future directions.

II. Wavelets for Texture Analysis

It is widely accepted that the solution to many problems in image coding and analysis becomes viable and even simpler when the image is transformed from the spatial domain to some other domain (e.g., Fourier domain). Wavelet transform has taken over well-established Fourier transform in many areas (e.g., image compression) because it provides both spatial and frequency characteristics of an image in the wavelet domain. The inherent potential of wavelets for their use in texture analysis was highlighted by [9], [10], which are amongst the earlier efforts in this direction.

A. Wavelets

Unlike Fourier transform, which employs sinusoids of varying frequency and amplitude as its basis functions, wavelet transform is based on small wave functions called wavelets of varying frequency and duration. It decomposes a 1-D signal $f(x)$ onto a basis of wavelet functions:

$$(W_a f)(b) = \int f(x)\psi^*_{a,b}(x)dx \tag{1}$$

where the basis functions $\psi_{a,b}(x)$ are obtained by translation and dilation of a single mother wavelet ψ:

$$\psi_{a,b}(x) = \frac{1}{\sqrt{a}}\psi(\frac{x-b}{a}). \tag{2}$$

In the above equation, a and b are the scale and translation parameters, respectively. The mother wavelet ψ is localized both in spatial and frequency domains.

B. Wavelet-based feature extraction

The fundamental idea behind filtering based feature extraction is to pass the image $I(x, y)$ through a bank of wavelet filters, observe their response $W(x, y)$, and compute some energy measure from these subbands. This is based upon the hypothesis that distinct textures of an image will have different subband energy measures which can be potentially useful for segmentation and classification purposes. For all our experiments, we used Daubechies' 8-tap filters to compute the wavelet subbands. The procedure for wavelet-based feature extraction is depicted in Fig. 1. In this setting, n decomposed subbands (excluding the DC subband) are reverse filtered to generate feature images $F_i(x, y)$. A local energy function is then computed corresponding to every pixel in each of the feature images yielding an $n-$dimensional feature vector which is assumed to have texture discriminant characteristics for segmentation or classification.

III. Ant Tree Clustering

Ants are able to build tree-like structures due to their self-assembling behavior. These types of self-assembly behaviors have been observed with Linepithema humles Argentina ants and African ants of gender Oecophylla longinoda. This ability has been recently experimentally demonstrated in [7]. Ants exhibit self-assembling behavior by building the tree-like structure from a fixed support, which could be a stem

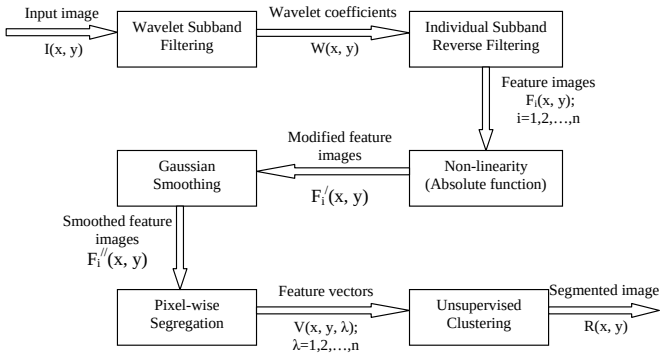


Fig. 1. Texture segmentation algorithm

or a leaf. They may move on the structure in order to be connected. This behavior has been used in [8] to estimate the number of clusters in a given data set. Our work concentrates on the probabilistic approach of ants towards getting them connected to the tree and iterative adjustment of the partitions, as mentioned in the future directions in [8]. In [8], a number of rules have been defined based on the self-assembling behavior. These rules alongwith a few problem-specific additional rules can be used to define artificial ants, which build trees to partition the data (see Fig. 2). The rules are:

1) Ants maintain a number of outgoing links towards other ants.
2) Ants maintain an incoming link from other ants.
3) Ants carry feature vectors containing characteristic information about which of the partitions they should belong to.
4) Ants maintain and update threshold values for similarity and dissimilarity.
5) Ants maintain their current position in the tree.
6) Ants may remain disconnected throughout the whole training process.
7) The global scope of similarity for ants should be adjusted iteratively. This global scope remains fixed for all the ants throughout an iteration of data partitioning. A suitable selection of this scope leads to a better partitioning of data.

Artificial ants capable of satisfying these requirements are designed which build ant trees.

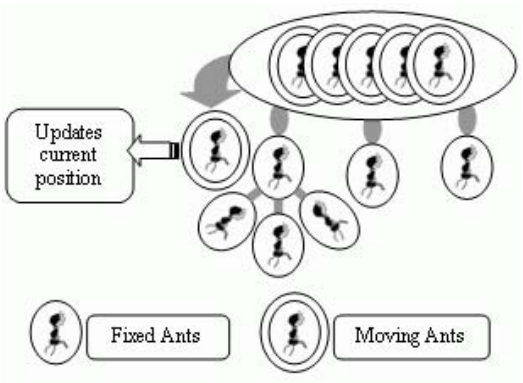


Fig. 2. General principles of tree building

A. Threshold Update Rules

Every ant maintains similarity and dissimilarity threshold values. During the ant tree construction, ants may fail to connect themselves to the tree. In case of such failure, the updating of threshold allows the ants to be more tolerant and increases their probability of being connected to the tree during the next iteration. Following are the similarity and dissimilarity updating rules:

$$\beta_{sim}(i) \leftarrow \beta_{sim}(i) \times 0.9 \tag{3}$$

$$\beta_{dissim}(i) \leftarrow \beta_{dissim}(i) + 0.01 \tag{4}$$

where i represents the ants whose threshold values are being updated. The values for β_{sim} and β_{dissim} are initialized as $\rho/2$ and 0 respectively, where ρ is the similarity scope. The effect of these threshold updates is discussed in the following sections.

B. Similarity Estimation

Similarity estimation depends on the nature of the problem. In case the data is in the form of vectors, simple Euclidean distance can be used. Every ant carrying data examines its similarity with all immediate children already fixed to the root of the tree. Based on the estimated similarity value, the ants direct themselves in the tree. The similarity measure $\xi(i,j)$ between ants i and j is calculated as follows:

$$\xi(i,j) = \rho - \sqrt{\frac{1}{M}\left(\sum_{k=1}^{M}(v_{ik} - v_{jk})^2\right)} \tag{5}$$

where M is the dimensionality of feature vectors v and ρ is the similarity scope value used to allow an ant j to consider all the ants that lie in the range of ρ units as similar ants which it can be grouped with. This is elaborated in Fig. 3 for ants carrying 2-D vectors.

Ant i may find all the immediate children ants lying in three different types of regions. These regions are different for each ant and are based on their similarity and dissimilarity threshold values (see Figs. 3 and 4). Ants (vectors) lying outside the scope of the current ant i are considered dissimilar ants. Ants that lie within the similarity scope are further partitioned based on the similarity threshold value of the current ant.

The similarity threshold value for each ant as mentioned before is initialized as $\rho/2$. However, this value may change during the segmentation process by using the similarity and dissimilarity update rules. The initialization value of $\rho/2$ for ants plays an important role in the segmentation process. Because of this, the initial least distortion between any two ants directly connected to the root would be $\rho/2$ units provided ρ is initialized with a very low value relative to the spread of the data vectors. This way the vector space can be partitioned in reasonable blocks based on the ρ value only. This is illustrated in Fig. 4 for ants carrying 2-D vectors. However, a suitable guess for the value of ρ is required in order to correctly partition the data. The larger the value of ρ, the fewer the number of partitions would be and vice versa. This is further discussed in the following section.

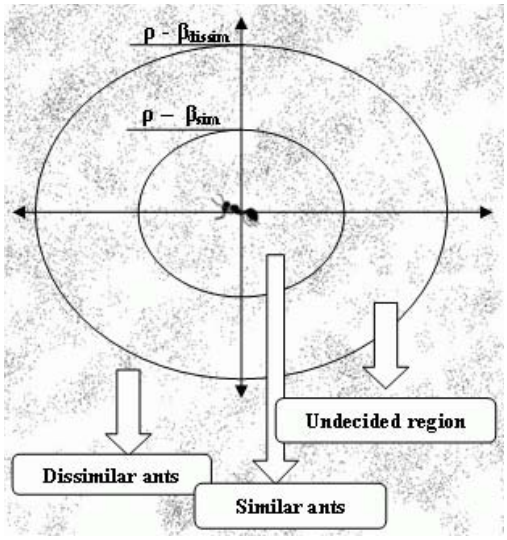


Fig. 3. Similarity scope for ants

C. Ant Tree Construction

Each ant starting from root of the tree examines its similarity with the ants that are already connected to the root. If no ants are connected to the root, then the ant connects itself to the root. If the root has some ants already connected to it, then the current ant finds the most similar ant connected to the root. Following rules are used to define similar and dissimilar ants:

1) If $\xi(i,j) > \beta_{sim}(i)$, then ant i is similar to ant j.
2) If $\xi(i,j) < \beta_{dissim}(i)$, then ant i is dissimilar to ant j.
3) If $\xi(i,j) \geq \beta_{dissim}(i)$ and $\xi(i,j) \leq \beta_{sim}(i)$, then ant i is neither similar nor dissimilar to ant j and needs further processing.

In the above rules, i represents the current ant and j represents the most similar ant connected to the root. If ant i ends up being dissimilar to ant j, then it attaches itself to the root. If the root cannot accommodate further ants, then ant i is dropped. Ant i moves itself in the direction of ant j if it ends up being similar and changes it position from root to the ant j. This is the second case. During the next iteration, ant i examines its similarity with the children of ant j and repeats the same process till it finds a suitable place. It should be noted that once ant i has moved from the root, it cannot be dropped. If it ends up being dissimilar to all the children ants of its current location, then it moves itself in the direction of the most similar child. For the third case, if ant i is neither similar nor dissimilar to the ant j, then it updates its similarity and dissimilarity threshold values using (3) and (4). While ant i updates its threshold values, other ants can proceed to build the tree. Due to this skipping of ant i and all the ants in the undecided region for further processing, only those ants connect themselves to the root of the tree which fall in dissimilar region. This effect is demonstrated in Fig. 4 for a 2-D feature space. Due to the threshold updating, if ant i was only marginally short from being similar to ant j, then the probability of ant i being connected to ant j increases the next time ant i gets its turn. Whereas, if ant i was marginally short from being dissimilar to ant j then the probability of

its being dissimilar to ant j increases. This is illustrated in Fig. 4 for ants carrying two dimensional feature vectors. The process continues till all the ants are either dropped or have fixed themselves to the tree. Each child of root alongwith its subsequent children is grouped together to form partitions. The centroids of these partitions are used to segment the textured images.

D. Complexity of the Algorithm

The similarity calculation is the key operation of the proposed clustering algorithm. Every ant needs to find its similarity with the ants already attached to the tree. Let us consider a scenario where the ants have just formed a complete binary tree, for a two-textured image, of depth d. The number of ants attached to the root of the tree is $2^{d+1}-2$. At this stage, any ant which needs to connect itself to the tree will require a minimum of $2d$ similarity calculations. Due to the probabilistic nature of the algorithm, the number of similarity calculations is greatly increased and cannot be exactly predicted. However, the above is a very rare case and suitable places can be found at any level of the tree. Due to the partitioning of the feature vectors based on the similarity scope, the number of iterations required to converge is reduced if the algorithm is used iteratively for the convergence of centroids. In some cases, this may give the algorithm a computational edge over the *k*-means clustering algorithm. Such an experimental setup is discussed in the following section.

The formation of a tree like structure increases the space complexity of the ant clustering algorithm as compared to the *k*-means algorithm. Every ant in the tree carries pointers to its parent and children nodes. Each ant also carries a feature vector, its similarity and dissimilarity threhsold values, its position in the tree, and position of the most similiar ant found in its neighbourhood.

IV. ATC - Experimental Setup

The value of similarity scope ρ and the order of ants play an important role in resulting partitions of the data. A number of segmentation strategies can be formed using different values for ρ and the order of ants. Following two were used for experimentation:

1) Random selection of ants for ant tree construction using some predefined value of similarity scope ρ, and
2) Adjustment of similarity scope ρ followed by adjustment of partitions.

Sorting of ants in either of ascending or descending order can also be used at the cost of extra computation. Ant trees were constructed using a random order of the ants. The process was repeated 50 times. Out of these 50 iterations, best, worst and average percent segmentation errors were calculated. The results are shown in Table I, which demonstrates the potential of algorithm to perform better than or comparable to the traditional *k*-means clustering algorithm.

The selection of ρ value also plays an important role. Since a suitable value for ρ cannot always be predicted, two rules are defined to adjust the value of ρ. The adjustment rules require a fixed number of desired partitions τ. The number of children that any node in a tree can have is also fixed to τ. If ρ is too small, most of the partitions will be smaller and the tree will quickly reach the maximum number of children limit while most of the ants still being disconnected to the tree. This situation is deliberately created by initializing ρ with a very low value. Because of this low value, most of ants end up being dissimilar to the ants already attached to the root of the tree. Consequently, most of the ants connect themselves to the root of the tree. The tree reaches the maximum number of children τ, which is predefined, in a fast manner. That is why the remaining ants cannot attach themselves to the tree if they are not similar to any of the ants already attached to the root and thus remain disconnected. The number of ants being dropped is maintained and is represented by λ. The value of ρ is updated in the next iteration to allow maximum number of ants. The algorithm uses the following adjustment rule:

$$\rho = \rho + \frac{\lambda}{\tau} \times \alpha \tag{6}$$

where λ is the number of ants dropped, α is a constant value, and τ is the maximum number of children allowed in the tree. However, a situation may arise where the number of partitions n is less than τ. To overcome this problem, another rule is defined:

$$\rho = \rho - \left(\frac{\tau - n}{\tau}\right) \times \alpha \tag{7}$$

where n is the number of resulting partitions. Equations (6) and (7) together help in iteratively adjusting the value of ρ to a suitable value.

The process is repeated until the number of ants being dropped goes below the permissible number of dropped ants. Once this happens, the second phase of centroids adjustment starts. This is also an iterative phase. Each child of the root node alongwith its subsequent children is grouped together and their centroid is calculated. After every iteration, the centroids of previously obtained partitions are used as initialization vectors for the next iteration of ant tree construction. These initialization vectors are carried by the artificial ants and are connected to the tree before the rest of ants. If required, the ρ value is also adjusted after every iteration. The algorithm stops when the difference between centroids of two consecutive iterations is very small.

V. Experimental Results

We experimented with 2- & 5-texture images generated through Brodatz textures [11] and MIT's VisTex album [12] either formed by ourselves or taken from Randen and Husoy [3]. Here we present our results using wavelet features and *k*-means or ATC based feature clustering. Fig. 6 shows the sample texture images containing both regular and irregular boundaries. Most of these textures can be found in natural objects such as wooden surfaces, sands, reptiles, grass etc. Experiments indicate that images with irregular boundaries between the texture regions are more complex to segment than the ones with texture regions having regular boundaries. It appears that this is also affected by the sensitivity of

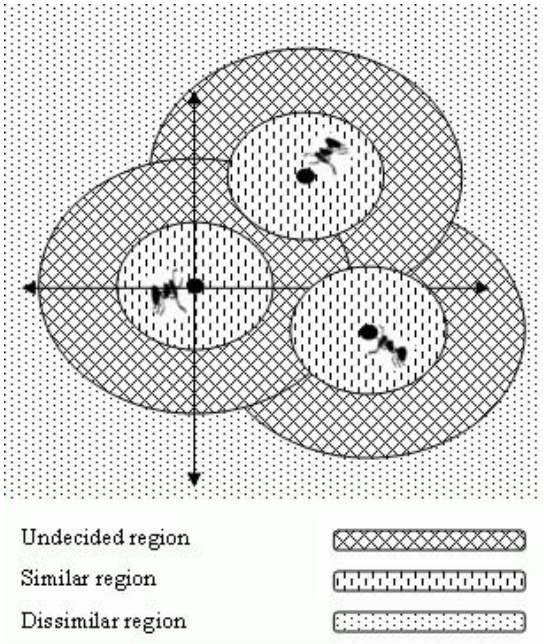


Fig. 4. Partitioning of vector space based on similarity scope ρ and initial similarity threshold value $\rho/2$ of ants

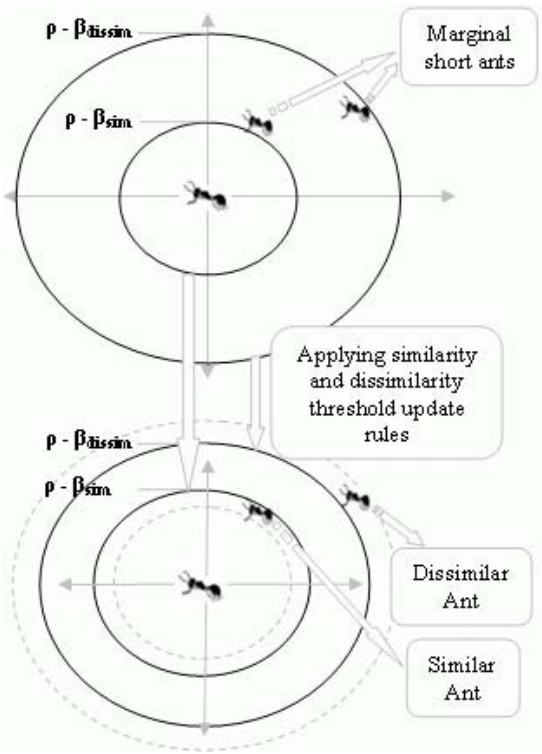


Fig. 5. Effect of applying similarity and dissimilarity threshold for 2-D feature vectors

wavelet based feature extraction to regular boundary shapes (horizontal, vertical, and diagonal).

It was observed from the visual results, not included here due to the limited space, that the two clustering algorithms are comparable and often have difficulties in similar areas. This might be due to the fact that in such areas, the algorithms do not have sufficient characteristic information from the wavelet features. Table II presents quantitative results for texture segmentation error on different texture images. It is evident that in majority of the cases, ATC outperforms *k*-means clustering. In addition, in Table II, these results are shown to compare favorably to the classification results given by [3] using a variety of filtering approaches for feature extraction on the same images.

VI. Conclusions

In this paper, we have presented a novel texture segmentation algorithm based on ant tree clustering of wavelet features. The clustering process is motivated by the self-assembly behavior of natural ants for forming mechanical structures such as drops, crossing chains, and building chains. Experimental results for segmenting multi-texture images reveal the promise carried by the proposed segmentation algorithm. While being computationally efficient, the probabilistic nature of the ant tree clustering algorithm allows it to avoid local minima which the traditional *k*-means algorithm may get stuck in. Our future work will attempt to improve the discrimination ability of features and the clustering algorithm's performance by introducing robust measures for construction of ant trees.

References

[1] K. Laws, *Textured Image Segmentation*, PhD thesis, University of Southern California, USA, 1980.

[2] M. Tuceryan and A. Jain, *Handbook of Pattern Recognition and Computer Vision, in Chen, C. and Pau, L. and Wang, P. (Editors)*, World Scientific, 2nd edition, 1998.

[3] T. Randen and J. Husoy, "Filtering for texture classification: A comparative study", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, pp. 1542–1550, 2002.

[4] E. Bonabeau, M. Dorigo, and T. Theraulaz, *From Natural to Artificial Swarm Intelligence*, Oxford University Press, New York, 1999.

[5] N. Franks and A. Sendova-Franks, "Brood sorting by ants: Distributing the workload over work surface", *Behav. Ecol. Sociobiol*, vol. 30, pp. 109–123, 1992.

[6] M. Dorigo and L. Gambardella, "Ant colonies for the traveling salesman problem", *BioSystems*, vol. 43, pp. 73–81, 1997.

[7] G. Theraulaz, E. Bonabeau, C. Sauwens, J-L. Deneubourg, A. Lioni, F. Libert, L. Passera, and R-V. Sol, "Model of droplet formation and dynamics in the argentine ant (linepithema humile mayr)", *Bulletin of Methematical Biology*, vol. 63, pp. 1079–1093, 2001.

[8] H. Azzag, N. Monmarch, M. Slimane, G. Venturini, and C. Guinot, "Anttree: a new model for clustering with artificial ants", in *IEEE Congress on Evolutionary Computation*, Australia, Dec 2003, vol. 1, pp. 2642–2647.

[9] M. Unser, "Texture classification and segmentation using wavelet frames", *IEEE Transactions on Image Processing*, vol. 4, no. 11, pp. 1549–1560, 1995.

[10] A. Laine and J. Fan, "Texture classification by wavelet packet signatures", *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 15, no. 11, pp. 1186–1190, 1993.

[11] P. Brodatz, *A Photographic Album for Artists and Designers*, Dover, New York, 1966.

[12] MIT, "Mit vision and modeling group", 1998.

TABLE I

SEGMENTATION ERROR (%) FOR 2−TEXTURE IMAGES USING RANDOM ORDER OF ANTS. SIMILARITY SCOPE FOR D9D19ER AND F17D9F IS 6.5 AND 6.25, RESPECTIVELY.

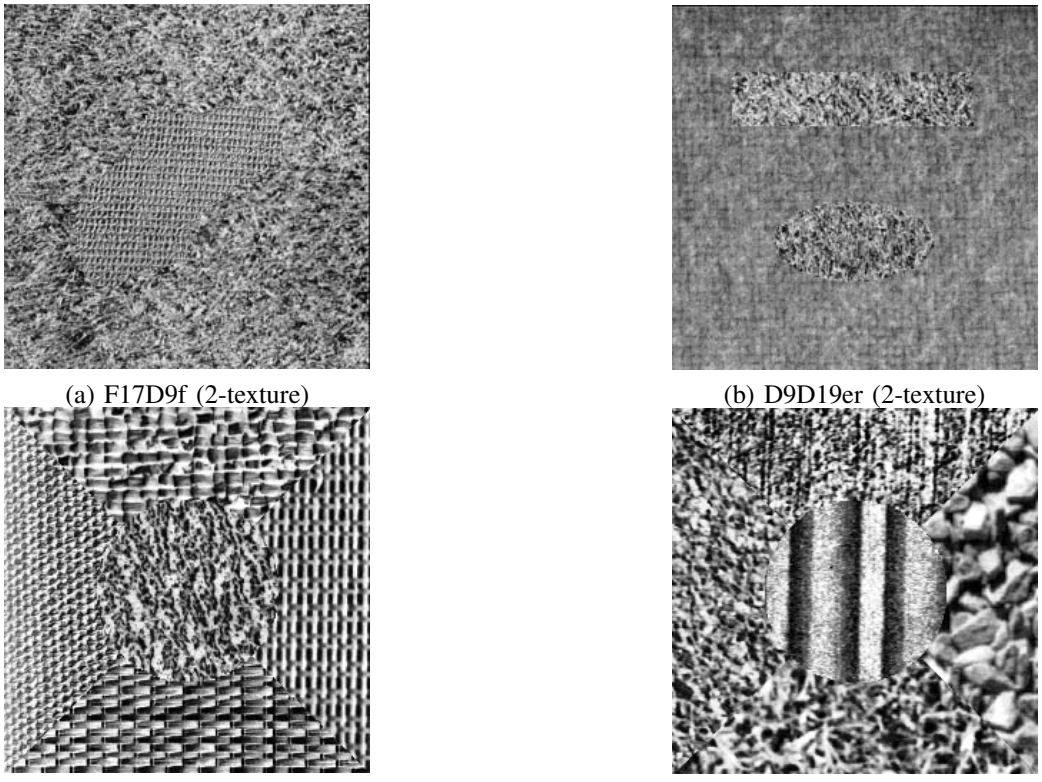
Image	*k*-means	ATC	Best	Worst	Average
D9D19er	1.89	Run 1 (50 iterations)	1.86	40.56	3.89
		Run 2 (50 iterations)	1.86	15.70	2.90
		Run 3 (50 iterations)	1.85	31.01	3.55
F17D9f	1.57	Run 1 (50 iterations)	1.56	20.95	2.43
		Run 2 (50 iterations)	1.57	18.61	2.54
		Run 3 (50 iterations)	1.57	35.82	4.03

TABLE II

SEGMENTATION ERROR (%) FOR 5−TEXTURE RANDEN & HUSOY IMAGES

Image	*k*-means	ATC	Average1*	Average2*	Average3*
Nat-5c	14.06	13.72	11.50	24.10	17.20
Nat-5v	25.43	25.12	27.20	44.00	33.60
Nat-5v2	37.55	36.62	25.90	38.70	33.50
Nat-5v3	43.99	45.16	31.10	41.00	33.20
Nat-5m	18.10	19.95	24.70	41.10	30.30

* [Average1, Average2, Average3] denote the average classification error taken from [3] using different mixtures of filtering techniques as follows:
Average1 - Heuristically designed texture feature extractors
Average2 - Optimized texture feature extractor
Average3 - Wavelet, Gabor, and QMF full-rate filter bank



(a) F17D9f (2-texture) (b) D9D19er (2-texture)

(c) Nat-5c (5-texture) (d) Nat-5m (5-texture)

Fig. 6. A sample of manually formed and Randen & Husoy $n-$texture images