A Genetic Programming Approach for Classification of Textures Based on Wavelet Analysis

Zheng Chen and Siwei Lu
Department of Computer Science, Memorial University of Newfoundland
St. John's, A1B 3X5, NL, Canada
{zchen, swlu}@cs.mun.ca

Abstract **– In this paper, we propose a method for classifying textures using Genetic Programming (GP). Texture features are extracted from the energy of subimages of the wavelet decomposition. The GP is then used to evolve rules, which are arithmetic combinations of energy features, to identify whether a texture image belongs to certain class. Instead of using only one rule to discriminate the samples, a set of rules are used to perform the prediction by applying the majority voting technique. In our experiment results based on Brodatz dataset, the proposed method has achieved 99.6% test accuracy on an average. In addition, the experiment results also show that classification rules generated by this approach are robust to some noises on textures.**

Keywords **– Classification, Genetic Programming, Texture Analysis, Wavelet Decomposition**

I. Introduction

Texture analysis is one of the fundamental issues in image processing and machine vision. It has been applied in a variety of fields, such as remote sensing, automatic inspection and medical image processing. The main problems of texture analysis are the texture classification, segmentation and synthesis based on texture features, which are believed to be the distinct characteristics of textures. The texture feature can be defined as a value, computed from the image of an object, that quantifies some characteristics of gray-level variation within the object [1]. To extract the features from images, lots of methods have been proposed based on perception of textures [2]. The statistical methods measure the spatial distribution of gray levels of image pixels, including the standard deviation and co-occurrence matrix-based features. The structure methods describe the texture as a combination of texture primitives. The spectral methods obtain the spectrum features by performing the Fourier transform. The model-based methods, such as Random Fields Models and Fractal Models, not only describe texture but also synthesize it using a specific model.

Genetic Programming (GP) [4], which is an evolutionary method that lends itself naturally to the development of program with the ability of automatically constructing appropriate structures for the solution as well as selecting the variables, has been applied to extract features from textures in the past. A GP method is proposed based on raw pixel data using dynamic range selection [5][6]. Gray level histogram, which is reported to outperform raw pixel features in the cluster accuracy and convergence speed, is also used as features to evolve the classification rules [7].

Inspired by the psycho visual study, multi-scale processing is a promising way to deal with texture analysis. Therefore multiresolution technique is applied to transform an image into a representation that can represent the spatial as well as the frequency information in different scales. Due to the well established formal and solid representation of wavelet, the multiresolution analysis obtains features from subimages of wavelet decomposition of 2D images.

In this paper, the energy features are extracted by Haar wavelet decomposition and then GP is applied to evolve rules to identify whether a texture image belongs to a certain class. The GP is able to produce classification rules that are robust to some random noises and achieves high test accuracy.

II. Extract Features Using Wavelet Decomposition

The main principle of Haar wavelet transform is decomposing a signal into components of basis wavelet functions:

$$\psi_{a,b}(x) = \sqrt{2^a}\,\psi(2^a t - b),$$
$$b = 0,1,\ldots \text{ and } a = 0,1,\ldots,2^j - 1,$$

which are the translation and scaling of a single prototype function:

$$\psi(x)=\begin{cases}1 & 0\le x<0.5\\ -1 & 0.5\le x<1\\ 0 & otherwise\end{cases}.$$

These components contain the information both in the frequency and time (space) domain. To obtain the wavelet decomposition of 2D image, the two-dimensional discrete wavelet transform (DWT) is used by applying the subband coding filters consecutively on horizontal and vertical directions.

The first level DWT decomposes the image into 4 subimages: the scaling (LL), horizontal (LH), vertical (HL), and diagonal (HH) subimage respectively. The scaling subimage is then decomposed iteratively, until the scaling subimage only contains one pixel. It is easy to known that for an NxN image, a maximum number of logN levels decomposition can be performed. This decomposition is also called pyramidal decomposition (Fig. 1).

For convenience of further study, we define that in the logN level, the scaling subimage is the 0th subimage and the horizontal, vertical, and diagonal subimages are the 1st, 2nd, and 3rd subimages respectively. Then for the nth level, the indices of horizontal, vertical, and diagonal subimages are defined as 3(L-n)+1, 3(L-n)+2, and 3(L-n)+3 respectively (Fig. 2).

Therefore, the pyramidal decomposition of an NxN image consists of 3logN+1 subimages, which contains spatial and frequency information in different scales.

The texture features then can be obtained by calculating the energy, which is the square sum of coefficients in subimage. For example, if the subimage is $f(x,y)$, where $0\le x\le M$ and $0\le y\le N$, then the energy of this subimage E is defined as:

$$E=\frac{1}{MN}\sum_{x=0}^{M}\sum_{y=0}^{N}f(x,y)^2$$

The energy is widely used because it is sound and has an obvious physical interpretation. The feature set now consists of energies of different scales, which is an important characteristic for texture analysis [3].

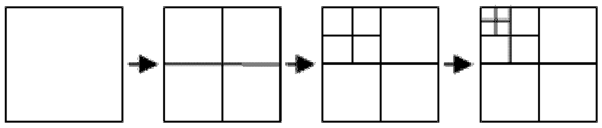
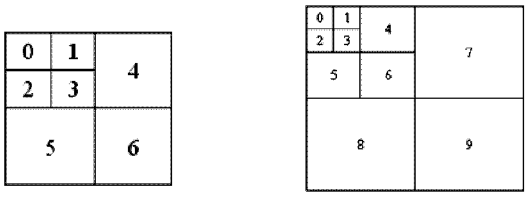


Fig. 1. Pyramidal wavelet decomposition



(a) 4x4 image (b) 8x8 image

Fig. 2. Index of subimages

III. Genetic Programming

A. Evolutionary process model

The Genetic Programming, an extension of genetic algorithm which evolves computer programs in the form of variable length parse tree as the knowledge model, is an evolutionary progress. In the initial stage, GP creates its own population that contains randomly generated programs, also called individuals. During the evolutionary progress, individuals in population evolve in parallel over generations based on Neo Darwinism, where the reproduction, inheritance, and variation describe that the offspring programs can be created by directly copying of parent programs to next generation, by randomly exchanging parts of parents programs, or by slightly modifying parents programs. The competition in GP is described by fitness based selection methods and the replacement strategy. The typical implementation of GP to evolve a computer program can be summarized as follows:

1) Initialize the population with randomly generated individuals.
2) Repeat the following steps until the termination criterion is satisfied:
   a) Until the new population is filled with offspring keep on doing following steps:
      i) Select two parents from the population with the probability based on fitness.
      ii) Perform genetic operators, such as crossover and mutation, on the selected parents to create new individuals (offspring) and then insert the offspring into new population.
   b) Replace the worst individual(s) in new population with the best individual(s) in the parental population.
3) Fetch the best individual in the population as the solution of the problem.

The following parts will focus on how to use GP to solve the texture classification problem. We will present the representation of classification rules, the fitness evaluation, and other configurations in GP.

B. Rule Representation

In texture classification problems, programs in GP are corresponding to the classification rules. A typical example of these rules is the following form:

IF (S - *expression*) $\ge$ 0 THEN 'Class A' ELSE 'Not Class A'

As a matter of fact, arithmetic S-expression for example: $(E4 - E9)\times(E4 - E8)$, is represented as a parse tree in GP (Fig. 3), where the leaves of the tree are referred as terminals and non-leaf nodes are referred as functions. The terminals in the above S-expression are E4, E8, and E9, which represent the energy values of the 4th, 8th, and 9th subimages respectively. In general, terminals can be both real

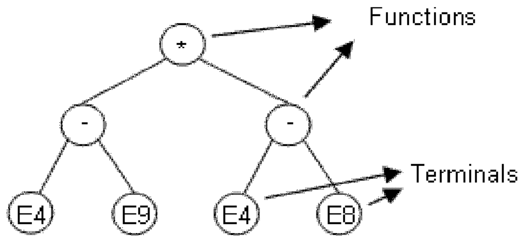


Fig. 3. The parse tree of S-expression

constant numbers and the feature value in the feature domain. The functions in the above S-expression only arithmetic operators * and – are used. However, in practice, functions can be set as either arithmetic operators such as {+, –, *, %, SQR, SQRT, EXP} or Boolean operators {AND, OR, NOR, NAND, XOR} or even both arithmetic and Boolean operators. Notice that during the calculation of expression, it is possible that the denominator of division can be 0. As a hard constraint, we treat the undefined result produced by denominator 0 as 1.

C. Fitness Evaluation

Fitness function, which evaluates the goodness of an individual, is one of the most important components in the evolutionary computation method, as well as in GP. In classification problem, the classification accuracy, which is ratio of the number of true predication and total number of samples, is usually used as the fitness function. A comprehensive method is proposed to calculate the fitness, which uses the area under the convex hull of the Receiver Operating Charctaristic (ROC) curve, as a fitness measure [9]. The ROC shows tradeoff between missing positive cases and raising false alarms. In this project, for the speed and simplicity, only one single threshold point on the curve was used, which leads to the fitness function:

$$fitness = \frac{1}{2}\left(\frac{N_{tp}}{N_p} + \frac{N_{tn}}{N_n}\right),$$

where N_{tp}, N_{tn}, N_p, N_n are the number of true positive prediction, the number of true negative predication, the number positive samples, and the number of negative samples respectively. It is clear that the rules which classify all the samples correctly achieve a fitness value of 1 and the ones which misclassify all the samples obtain a fitness value of 0.

D. Rules for Classification

In the binary class problem, instead of using only one rule for the class predication, multiple rules can be assembled together to determine the class labels. Among a variety of methods that assemble rules, majority voting is one of the most intuitive and efficient ones. A successful use of majority voting technique on the gene expressions classification problem based on GP is reported [8]. During predication, the samples are tested on N rules respectively. The class label of a certain sample is determined by the prediction of majorities in rules. That is if more than half of the rules made a positive prediction then the given sample is treated as positive sample or otherwise. Table I gives an example of how majority voting works in binary classification, where five rules are used as a majority voting group.

In multi-class problem, it is not an easy task to map all the classes to the output of a single rule. Although in the work of [5] a dynamic range selection (DRS) method is presented that DRS is suitable for multi-class problems, the DRS is dependent on the output domain and it requires training additional samples to get the output segmentation. An alternative way is treating the multi-class problem as a combination of many binary classification problems, namely one vs. others strategy. It is used to make a class predication by applying binary classification on each class. Ideally, one sample should be reported as positive in only one class, and be negative in all others. However, in practice, it is a common phenomenon that the sample is reported as positive in more than two classes. In order to deal with this situation, majority voting is applied again. If a given sample X is tested on a group of rules, the number of the positive predication is recorded for each class. The class which obtains the highest positive count is viewed as the predicted class. If more than one rule has the same positive count, the class with lower index is assigned to be the predicted one.

E. Configuration of GP

We generate the initial population using the "ramped half-and-half" method with a maximum initial depth of 4. A proportional selection method is applied to select two parents from the population. The sub-tree crossover is used with the probability of 0.9. The terminal and function mutation are used with the probability of 0.1. The elitism is used to preserve the best individual in previous generation. Because shorter rules are more general than longer ones with the same accuracy and lower risk to be over fitting on the training set, it is advisable to evolve a rule as shorter as possible without losing the training accuracy. In our implementation, a length comparison happens when selecting the best individual as elitist, such that the fitness evaluation and genetic operators will not be affected. In order to refine an even shorter rule, the evolution progress will not terminate until 200 generation passed, no matter when the best rule achieves the fitness of 1. More GP parameters are list in Table II.

TABLE I.

MAJORITY VOTING FOR BINARY CLASSIFICATION

	Sample 1 (P)	*Sample 2 (P)*	...	*Sample n (N)*
Rule 1	P	N	…	N
Rule 2	P	N	…	P
Rule 3	N	N	…	N
Rule 4	P	N	…	P
Rule 5	P	N	…	P
Prediction	TP	TN	…	FP

TABLE II.

GP PARAMETERS USED IN EXPERIMENT

Parameter	*Value*
Population size:	200
Maximum depth:	6
Maximum generation:	200
Maximum initial depth:	4
Crossover probability:	0.9
Reproduction probability:	0.1
Mutation probability:	0.1
Elitism:	Yes

IV. EXPERIMENTS

Five types of textures (shown in Fig. 4) are obtained from the bench mark dataset: Brodatz texture album [10]. For each class, 20 non-overlapping samples with the size of 64x64 are randomly selected from original textures. Among them 10 ones are used to train the classifier and other 10 are used as testing samples. Specifically, using the pyramidal decomposition, 19 energy features can be obtained from each image.

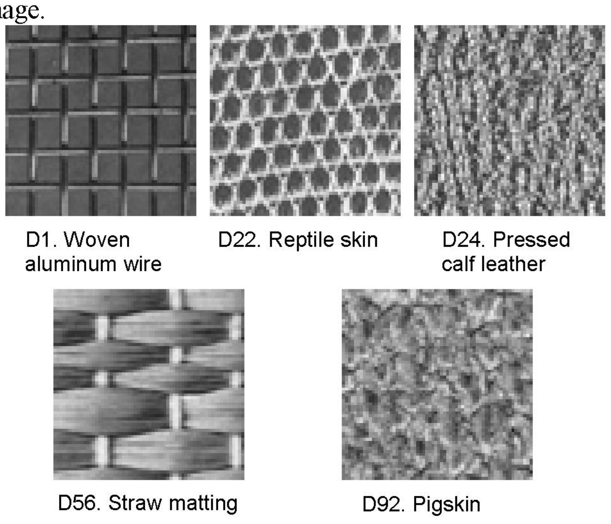


Fig. 4. Examples of texture samples

A. Perfect Classification Rules

To get the perfect classification rules, all the 100 (=5*20) samples are used to train, where the size of the majority voting group is set to be 10. A learning curve which gives the average fitness of these 10 rules over 200 generations is presented in Fig. 5. From the curve, we can find that the fitness of these rules in class 3 and 4 converge to the average fitness 1. That means any single rule evolved in class 3 and 4 can classify all the samples correctly. However, the average fitness of rules in class 1, 2, and 5 did not reach the value of 1, which indicates that not all the evolved rules successfully classify the training samples. However, majority voting groups of rules in class 1, 2 and 5 successfully predict the classes of all training samples. According to the curves we also find that the fitness convergence speed of rules in class 3 and 4 is much faster than the ones in other three classes. These results suggest that the classification rule for class 3 and 4 should be "easier" than the others.

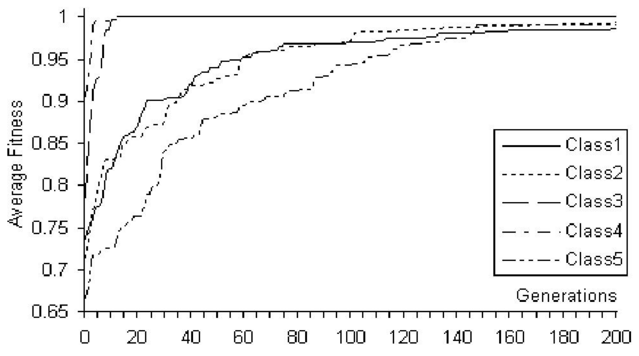


Fig. 5. The learning curve of rules

Table III gives some examples of perfect classification rules (one in each class). Note here that the rules in each majority voting group are not unique, namely some rules are the same. For example, the rule R_3: –(E18,E11) in class 3 has been added to the majority voting group in 102 out of 200 GP runs and R_4: –(E11,E16) in class 5 has been added in 112 out of 200 GP runs. Further more, although some of the rules have different structure, they actually have the same function. Such as the rule –(–(–(E12,E18),E9),E9) and –(–(–(E12,E9), E18),E9), both of them calculate the same expression E12–E18–2*E9. The average length of rules in class 3 and 4 is much shorter than the others. This again suggests that the classification problem that identifying textures in class 1, 2, and 5 should be "harder" or "more complicated" than the ones in 3 and 4, which leads to the result that more generations and more features are needed to construct the rules.

B. Test accuracy

To get the test accuracy, classification rules are evolved according to 10 training samples and other 10 samples are tested against the evolved rules. The test accuracy is obtained by using the number of correct prediction to divide the overall number of test samples. Notice that GP is a stochastic progress, thus we perform 20 runs of experiment, and then get the average accuracy. The results are shown in Fig. 6.

TABLE III.

MAJORITY VOTING FOR BINARY CLASSIFICATION

Rule	*Expression of rule*
R_1	+(–(E1,SQR(E15)),*(–(E16,E8),E9))
R_2	–(–(–(E15,E18),E9),E9)
R_3	–(E18,E11)
R_4	–(E11,E16)
R_5	+(E15,–(E18,–(E16,E18)))

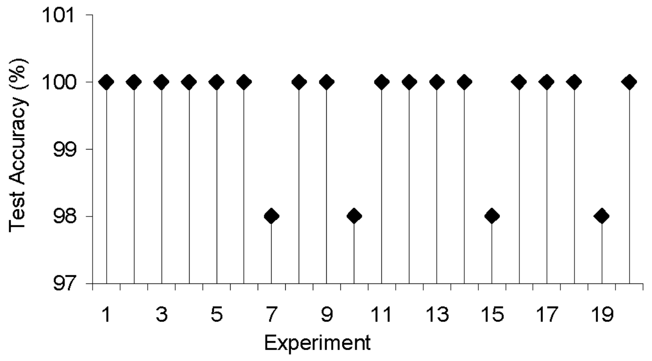
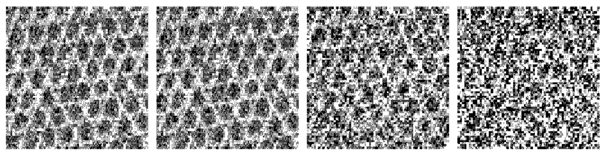


Fig. 6. Test accuracies in 20 runs

From the results, we can find that 16 out of 20 experiments achieved 100% accuracy. The other four experiments also got 98% accuracy, which means only one out of 50 samples is misclassified, and on average, the test accuracy is 99.6%. This is really a very encouraging result, which gives us the belief that the proposed method is suitable to be utilized in the field of texture classification.

C. Noise tolerance

A good texture classifier should be capable of correctly classifying the samples with noises to some extend. The sensitivity of the proposed method to noisy data is tested by adding the white Gaussian noises, on both training and testing samples before classification. Four experiments are tested to get the test accuracy, where the signal-to-noise ratios (SNR) of noisy samples are 1dB, 5dB, 10dB, and 15dB respectively. Some examples of noisy texture samples from D22 are shown in Fig. 7.



(a) 15dB (b) 10dB (c) 5dB (d) 1dB

Fig. 7. Examples of noisy texture samples (From D22)

The results are summarized in Fig. 8, where the average, maximum, and minimum values of test accuracies in 20 runs are compared. We can find that when the SNR is 10dB, the test accuracy still can achieve a reasonable value of 95.5%. When the SNR is lower than 10dB, the test accuracy is decreased dramatically. We also find that when the SNR is higher than 10db, most of the perfect classification rules that classify the noisy samples correctly are the same as the ones evolved using original samples. Further more, when the SNR is 1dB, none of perfect classification rules in class 1, 2, 3, and 5 can be evolved. In fact, for those samples with the SNR of 1dB, it is too noisy to tell the differences even with human eyes. However, samples in class 4 maintain some features which are noticeable to human eyes.

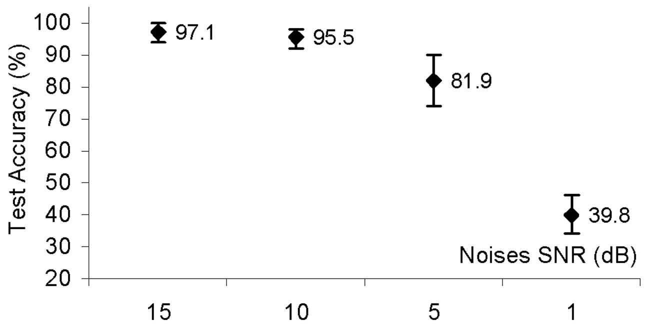


Fig. 8. Test accuracies on different noises

V. Discussions

A. Tree-structure decomposition

The traditional pyramid-type wavelet transform recursively decomposes subsignals in the low frequency channels. However, since the most significant information of a texture often appears in middle frequency channels, performing further decomposition only in the lower frequency region, such as in the conventional wavelet transform, may not help much for the purpose of classification [11]. An alternative method, called tree-structure decomposition, is used to further decompose detailed subimages in pyramidal wavelet decomposition. It is obviously that tree-structure decomposition provides much more features than the former one. With more features, detailed information is available to the classifier. On the other hand it also dramatically increases the dimensionality of the class feature space, which leads to great increase in the difficulty of classification. However, with the belief that GP can automatically select informative features from large feature space, it becomes a promising way to use GP as the feature selector and classifier where tree-structure decomposition is applied.

B. Interpretation of rules

Comparing with other classifiers, one of the merits of using GP is that it is transparent in the sense that the mechanism used to classify samples is available for inspection. In the proposed approach, classification rules are arithmetic combination of energy features. Given a rule, say –(E21,E12), an interpretation can be concluded that this type of texture has much higher energy in the 21st subimage and lower energy in the 12th subimage. The same conclusion can be obtained by analyzing the positive/negative contribution of terminals. However, when it comes to the meaning of * and %, it becomes harder to interpret. Especially, in the situation where multiply two subtraction expression together, for example *(–(E21,E12),–(E14,E7)), where the positive/negative contribution is not fixed unless a certain sample is assigned.

Therefore to develop a comprehensible rule representation becomes an inevitable need for the GP classifier.

C. The size of majority voting group

Majority voting technique plays an important role in the proposed approach. It is clear that the performance of majority voting is closely related to the size of the ensemble of rules. A large number of rules should improve the prediction accuracy. However, it does not lead to the conclusion that the more rules used the higher accuracy can be achieved. Further investigation is needed to find the number of rules, which can give the optimal results.

VI. Conclusion

In this paper, we proposed a method that uses GP to evolve rules for the texture classification problem. Each evolved rule contains an S-expression, which is the arithmetic combination of the energy features of subimages in the pyramidal wavelet decomposition. A majority voting technique is also applied to enhance the performance of prediction and deal with the multi-class problem. A set of perfect rules, which classify all the samples correctly are obtained by the GP. Using these perfect rules, most parts of the noisy images generated from the training samples can be classified accurately. The test accuracy of 99.6% is achieved over 20 runs. All these results give us the confidence that the proposed method is suitable for the texture classification problem. However, as we discussed before there are many problems remained in the proposed method and further more efforts are still needed.

References

[1] Kenneth R. Castleman, *Digital Image Processing*, Second edition, Prentice Hall 1996.

[2] T. R. Reed and J. M. H. Du Buf, "A review of recent texture segmentation and feature extraction techniques," *Computer Vision Graphics Image Process: Image Understanding*, vol. 57, pp. 359-372. Mar. 1993

[3] S. Livens, P. Scheunders, G. Van de Wouwer, and D. Van Dyck, "Wavelets for texture analysis, an overview," *In Proc IPA*, IEE Pub, No. 443, vol. 1, pp. 581--585, 1997.

[4] Koza, J.R, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA, 1992.

[5] A. Song, T. Loveard, and V. Ciesielski, "Towards Genetic Programming for Texture Classification," In *Proceedings of the 14th Australian Joint Conference on Artificial intelligence: Advances in Artificial intelligence* (December 10 - 14, 2001), M. Stumptner, D. Corbett, and M. J. Brooks, Eds. Lecture Notes In Computer Science, vol. 2256. Springer-Verlag, London, pp .461-472.

[6] A. Song, V. Ciesielski and H.E Williams, "Texture classifiers generated by genetic programming," In *Evolutionary Computation, 2002, CEC '02, Proceedings of the 2002 Congress on*, vol. 1, pp. 243 -248, May 2002.

[7] B. Lam and V. Ciesielski, "Discovery of human competitive image texture feature extraction programs using genetic programming," In Kalyanmoy Deb et al., editor, *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO2004)*, vol. 2, pp. 1114-1125. Springer, June 2004.

[8] Topon Kumar Paul, Yoshihiko Hasegawa, and Hitoshi Iba, "Classification of gene expression data by majority voting genetic programming classifier," *in Proceedings of 2006 IEEE World Congress on Computational Intelligence*, Vancouver, BC, Canada, July 16-21, 2006, pp. 8690-8697.

[9] William B. Langdon and Bernard F. Buxton, "Evolving Receiver Operating Characteristics for Data Fusion," *EuroGP'2001*, LNCS 2038 pp. 87—96.

[10] Phil Brodatz, *Textures: A Photographic Album for Artists and Designers*, Dover, NY, 1966.

[11] Chang, T., Jay Kuo, C.-C., "Texture analysis and classification with tree-structured wavelet transform," *IEEE Trans. Image Process.* 2 (4), 429-440, 1993.