Applying genetic programming to learn spatial differences between textures using a translation invariant representation

Lam, Brian; Ciesielski, Victor

Lam, B., & Ciesielski, V. (2005). Applying genetic programming to learn spatial differences between textures using a translation invariant representation. 2005 IEEE Congress On Evolutionary Computation Proceedings, Vol. 3, 2202–2209. https://doi.org/10.1109/CEC.2005.1554968

Published Version: https://doi.org/10.1109/CEC.2005.1554968

Applying Genetic Programming to Learn Spatial Differences Between Textures Using A Translation Invariant Representation

Brian T. Lam
RMIT University
124 La Trobe Street
Melbourne, VIC 3000
blam@cs.rmit.edu.au

Vic Ciesielski
RMIT University
124 La Trobe Street
Melbourne, VIC 3000
vc@cs.rmit.edu.au

Abstract- This paper describes an approach to evolving texture feature extraction programs using tree based genetic programming. The programs are evolved from a learning set of 13 textures selected from the Brodatz database. In the evolutionary phase, texture images are first "binarised" to 256 grey levels. An encoding of the positions of the black pixels is used as the input to the evolved programs. A separate feature extraction program is evolved for each of the 256 grey levels. Fitness is measured by applying the evolved program to all of the images in the learning set, using one dimensional clustering on the outputs and then using the separation between the clusters as the fitness value. On two benchmark problems using the evolved programs for feature extraction and a nearest neighbour classifier, the evolved features gave test accuracies of 74.6% and 66.2% respectively for a 13 Brodatz and a 15 Vistex texture problem. This is better than a number of human derived methods on the same problems.

1 Introduction

Computer vision has many unsolved problems providing a fertile ground for the application of genetic programming. One of these problems is texture classification. While there is a considerable history of work on visual texture, the definition of texture is still imprecise. However, it is generally agreed that a texture is spatially homogeneous and contains repeated structures. In synthetic textures, such as horizontal lines, vertical lines (figure 2) or a checkerboard, the basic structure is repeated exactly. In natural textures, such as grass, wood, sand or rocks, there is some random variation in size, shape, intensity or colour in the repetitions of the basic structure. Figures 11 and 12 show some examples of natural textures.

The spatial difference between two textures can be considered as the different arrangement of pixels at each grey level within each repeated structure. Examples of spatial differences between vertical and horizontal lines are given in figure 4. In this simple case the spatial differences can be captured in a 4×4 window. Vertical and horizontal lines are synthetic textures in which a small basic pattern is repeated without change. In real world textures such as bark and brick (figure 10) the spatial differences cannot be captured in such a small window and, since the basic pattern is repeated with variations, the spatial differences will have a probabilistic nature; that is, patterns will apply to most, but not necessarily all of the examples of the texture. Examples of spatial differences between bark and brick are shown in figure 9.

Texture classification has been an ongoing research problem for many years. The goal of texture classification is to assign an unknown sample to one of several predefined categories. This is achieved by first extracting textural features from the images followed by training a classifier which can be used on unknown samples. The process is shown in figure 1. The performance of the classifier depends on the textural features extracted. Many texture feature extraction techniques have been proposed, they can be categorised as: statistical, geometrical, model-based or signal processing based[1]. Current texture feature extraction algorithms are derived by human intuition and analysis. This approach has a relatively low artificial to intelligence ratio [2], that is, most work is done by the human. In contrast, in this paper we explore the possibility of deriving such algorithms using genetic programming. This approach has a higher artificial to intelligence ratio [2], that is more work will be done by the artificial method and less intelligence is supplied by the human. Texture classification has been successfully applied in many areas, for example, remote sensing [3], automatic inspection[4], medical image processing [5] and document processing [6].

1.1 Related Work

Genetic programming has been applied to a variety of texture classification problems. The work so far can be grouped in to three approaches. The first approach involves preprocessing images with low-level texture feature extraction algorithms followed by genetic programming designed to discover the classification rule. The feature extraction algorithms are based on human developed theories and intuitions on what would be discriminating features over a wide range of different textures. Ross et al [7] used statistical features to classify microscopic images of minerals. Lin and Bhanu [8] used a co-evolutionary approach to build composite features from primitive features. Daida et al [9] evolved a classifier from texture features of SAR images. Howard and Roberts [10] evolved a vehicle detector from texture features of infrared images. This approach involves the steps of feature extraction, training and classification shown in Figure 1.

The second approach involves working directly from image pixels instead of using features. Song et al [11] developed a pixel based approach to classify grey scale texture

0-7803-9363-5/05/$20.00 ©2005 IEEE.

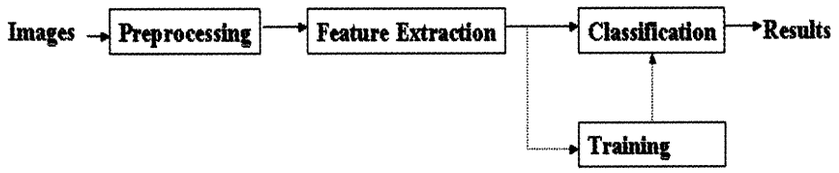


Figure 1: Image Texture Classification

images. This approach combines the feature extraction and classification steps shown in Figure 1 into a single step.

In the third approach, genetic programming is used to substitute for human development of generalised feature extraction algorithms. The evolved algorithms are expected to be applicable to a wide range of situations just like those developed by humans. The contrast between the conventional approach and this one is shown in figure 5. The focus of this paper is the boxed step (Feature Extraction Programs evolved by GP from Brodatz learning set) in the top right hand corner of figure 5. In other work using this approach, Koppen and Liu [12] evolved a texture detector to separate textured from non textured regions without priori knowledge of the image content. Kueblbeck [13] evolved Haralick features from the co-occurrence matrix. Lam and Ciesielski[14] used histograms as input to evolve 78 feature extraction programs from Brodatz textures. The evolved programs are then applied to both Vistex training and testing set. A nearest neighbour classifier applied to these data sets gave a test accuracy of 74.8% which was very competitive with the accuracy of human derived features on the same test set. However histograms do not capture any of the spatial information of textures and we expect that the accuracy of the evolved feature extraction programs would be increased if the spatial information was used. The use of texture spatial information, in the form of spatial differences, is addressed in this paper.

1.2 Research Questions

In this paper we propose a new feature extraction method based on learning the spatial differences between textures using genetic programming. Our contribution is a translation invariant representation at each grey level. Our research questions are:

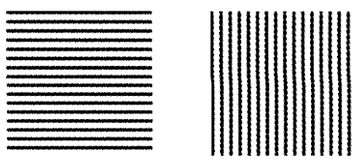
1. Can we use genetic programming to learn the spatial differences between textures?
2. How can the image data be encoded to facilitate the learning of spatial differences between textures?
3. How does the classification accuracy of a nearest neighbour classifier using textural features based on learning the spatial differences compare with accuracy of the nearest neighbour classifier using the textural features derived by human intuition?

For clarity of explanation, we present a simplified example of the spatial encoding and fitness evaluation on synthetic binary images in section 2. The extension to grey level textures is given in section 4.2. All of our experimental work is on grey level textures, examples of binary textures are used only to facilitate explanation.

2 Translation Invariant Encoding

In this section, we examine a simplified binary texture problem involving two synthetic black and white images with no noise in order to describe the inputs to the evolutionary process and how the evolutionary process can deliver highly discriminatory features.

Let us consider the simple example of two black and white textures, one with horizontal stripes and another with vertical stripes, as shown in figure 2. Each of the stripes is one pixel wide and the stripes are four pixels apart. If we were to randomly cut sub-images of size 4 x 4 from each, we would get eight possible patterns for the images, four for the horizontal, shown in the top row of figure 4, and four for the vertical, shown in the bottom row of figure 4. The encoding is as follows: Assume the positions are numbered from 1 to 16 in row major order as shown in figure 3. Let Pi give the pixel position corresponding to the ith black pixel. For example, in the bottom left picture of figure 2 the first black pixel $P1$ is in position 1 and the second black pixel $P2$ is in position 5. The full encoding of this image is $P1 = 1, P2 = 5, P3 = 9, P4 = 13$. There are only 4 black pixels so only P1-P4 are used. The encodings of the other images are also given in figure 4. Our objective is to find formulas that when applied to the positions of black pixels would give us two distinct numbers, one for the horizontal striped images and another for the vertical striped images, thus making the formulas highly discriminating for the two textures.



Horizontal Stripes Vertical Stripes

Figure 2: Horizontal and Vertical Stripe Binary Images

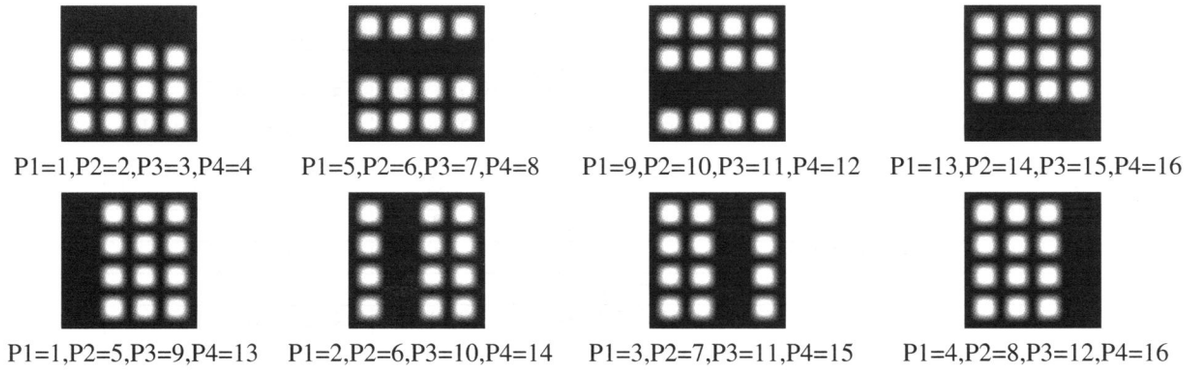


Figure 4: Encodings of horizontal and vertical stripe texture images

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Figure 3: Position Numbering

We will apply a few variations of possible formulas to the images and see if it is possible to get discriminating features. These are shown in table 1.

The examples in table 1 show that formulas 3 and 4 give distinct numbers for the vertical and horizontal textures. If we need to discriminate between horizontal and vertical striped images we need only evaluate formula 3. If formula 3 evaluates to -2, the image is horizontal stripes. If formula 3 evaluates to -8, then the image is vertical stripes. There could be many formulas that give discriminating features. Formula 4, for example, could also be used. The task for genetic programming is to discover one of these discriminating formulas. Requiring a single number is, in fact, too strict. If, for example, all of the outputs of a formula for texture 1 are less than zero and all of the outputs for texture 2 are greater than zero, the formula is just as discriminating. As long as there is no overlap between the two output ranges, a formula can be used to accurately discriminate the textures. For real world textures it is very unlikely that there will be no overlap between output ranges and it is necessary to think in terms of minimising overlaps. We develop this idea further in section 4.

3 Texture Data Sets

There are two texture libraries that are used in most of the research in texture analysis - the Brodatz album and the Vistex data set. Both contain images of natural textures with 256 grey levels. The Brodatz album consists of homogeneous categories of naturally occurring textures. The Vistex set consists of heterogeneous categories of texture images; that is, each class may have more than one type of texture. For example, the flower category may have flower images at three different resolutions, thus making the Vistex set more difficult to classify.

Conventional classification problems normally have a training and a testing data set. However for our experiment, we have an extra data set which we call the learning set. A *learning set* is the set of images used by the evolutionary process to evolve feature extraction programs. These programs are then used on a different *training set* of images to get a nearest neighbour classifier which is evaluated against a different *test set*. Figure 5 shows the difference between the conventional approach and ours. In the conventional approach as shown on the left hand side of figure 5, features are computed using human derived algorithms, a training set is used to learn the classifier, and a test set is used to estimate the true error rate. In our approach we use the learning set to evolve the feature extraction programs that will subsequently be used in the train-and-test stage. To test the generality of the evolved features we use Brodatz images for the learning set and Vistex images for the test and training sets.

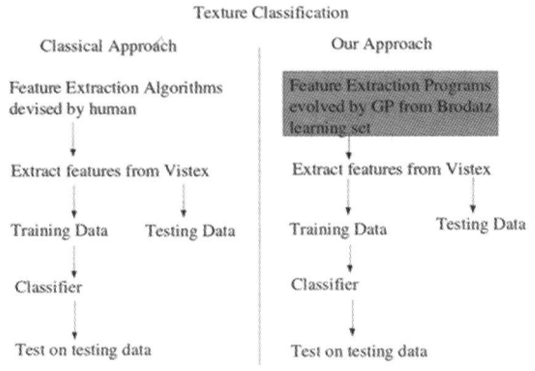


Figure 5: Texture Classification Approach

4 Confi guration of Genetic Programming

4.1 Selection of the Learning Set

The selection of the learning set is a major issue. If this is too small it will not be possible to learn texture regularities

Table 1: Variations of possible solution using plus and minus functions

Horizontal Stripes						
Input		1,2,3,4	5,6,7,8	9,10,11,12	13,14,15,16	
Formula No.	Formulas	Output	Output	Output	Output	Results
1	P1+P2+P3+P4	10	26	42	58	no good
2	P1-P2-P3-P4	-8	-16	-24	-32	no good
3	P1-P2+P3-P4	-2	-2	-2	-2	good
4	P1+P2-P3-P4	-4	-4	-4	-4	good
Vertical Stripes						
Input		1,5,9,13	2,6,10,14	3,7,11,15	4,8,12,16	
Formula No.	Formulas	Output	Output	Output	Output	Results
1	P1+P2+P3+P4	28	32	36	40	no good
2	P1-P2-P3-P4	-26	-28	-30	-32	no good
3	P1-P2+P3-P4	-8	-8	-8	-8	good
4	P1+P2-P3-P4	-16	-16	-16	-16	good

that will be useful in discriminating a wide range of other textures. If it is too large the computational requirements will be excessive. To test our approach, we have selected the same 13 Brodatz images as used in [14] to facilitate comparison. These images were chosen at random but with a view to maximising variety. We leave for future work the problem of selecting an optimal set of textures for the learning set.

4.2 Inputs

The size of the Brodatz texture images is 640×640 pixels. To create learning set we have randomly sampled 64×64 sub images. Ideally the inputs to the evolutionary process would be pixel intensities for each image. However this would result in programs with an enormous number of inputs. Also the search method would not focus on the spatial differences. To make the problem tractable we have done the following: (1) Selected 64×64 sub-images of 13 textures from the Brodatz library (2) For each sub-image, we have generated 256 binary images by extracting pixels at the same grey level into a binary image, one for each grey level from 0 to 255. Examples of binary images are shown in figure 9. (3) For each binary image, we have encoded the positions of the black pixels using the translation invariant encoding described in section 2. The process is summarised in figure 6. We have 256 problems like the one described in section 2 and their associated learning sets. After the completion of the evolutionary step there will be 256 feature extraction programs. The training and testing of the classifier will be done with all of these 256 features.

4.3 Functions

Originally we used the function set $\{+, -, *, /\}$. However we found that using just $+$ and $-$ gave feature extraction programs that were just as accurate as those using all four operators but were considerably easier to understand. Thus all subsequent work was carried out using just the $+$ and $-$ functions.

4.4 Fitness Evaluation

A feature extraction algorithm is considered useful if the feature values result in high classification accuracy. As described in section 2 this will occur if the feature values computed for each class are well separated in feature space. Thus, to evolve feature extraction algorithms, we need a way to implement the intuition that "the better the separation, the better the fitness". We have done this by computing the overlap between clusters generated by the k-means clustering algorithm.

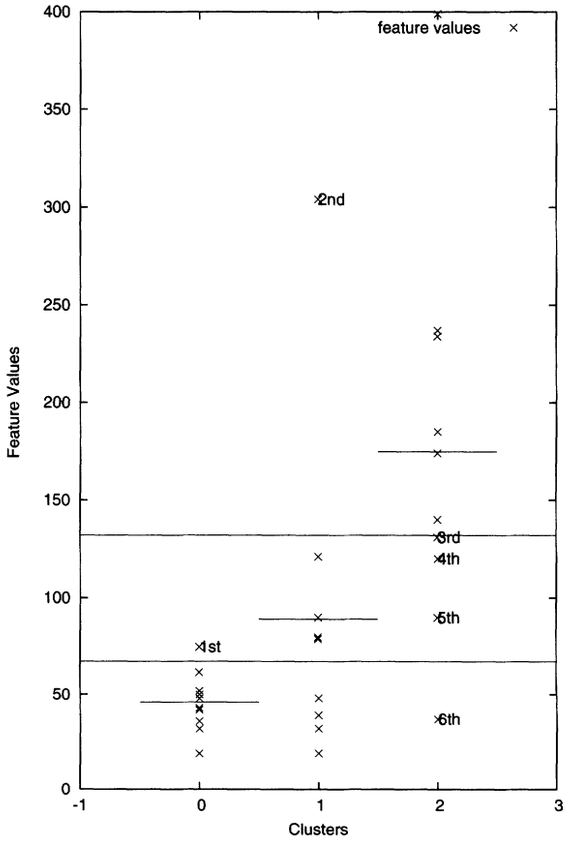


Figure 7: Feature Space for Three Texture Classes

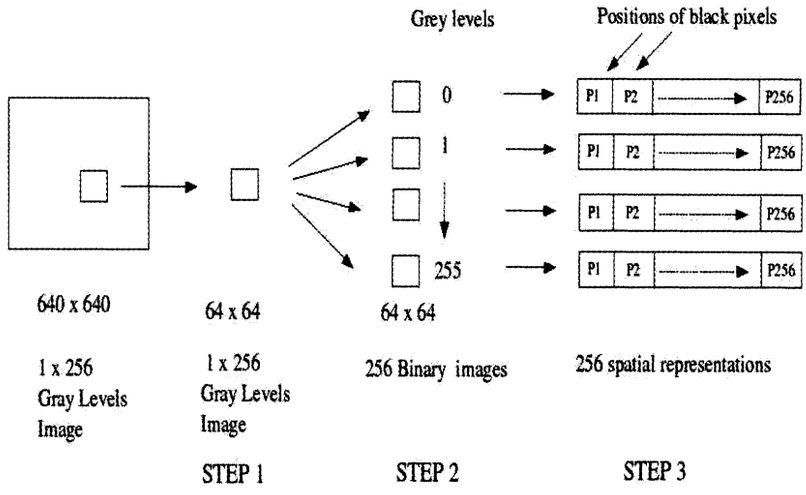


Figure 6: Spatial Representation

An example of a case for which there are three texture classes in the learning set is shown in figure 7. To get the data shown in figure 7 a program in the population has been evaluated on a learning set of 30 images which consist of 10 examples of texture 1, 10 examples from texture 2 and 10 examples from texture 3. The averages of the feature values for each texture give cluster centroids shown as short lines at 46, 89 and 175. The mid points between the first and the second pair of centroids, that is, the cluster boundaries are shown as long lines at 68 and 132. There is 1 cluster1 feature value above the first boundary line, 1 cluster2 feature value above the second boundary and 4 cluster3 feature values below the second boundary, thus 6 points are incorrectly clustered. Equivalently, it can be considered that there are 6 errors. This approach can be extended to the case of thirteen textures, where we try to minimise the overlap of thirteen clusters instead of three.

For this problem fitness evaluation is expensive. To evaluate an individual program requires loading all of the images in the learning set, applying the program to each image, saving the outputs, clustering them, and then computing the overlaps between 13 clusters.

4.5 Learning the Spatial Differences between Textures

As there are 256 spatial representations, one for each grey level, we can evolve 256 feature extraction programs. Our approach to learning the spatial differences of textures is summarised in figure 8. Each program generated is used to extract features from the spatial representations. The feature values are then clustered, the separation of the clusters is then used to rank the programs. This evolutionary process continues until the perfect separation between clusters is achieved or some predetermined number of generations is reached. The 256 evolved feature extraction programs are then used to extract 256 features for the train-and-test phase of the work.

4.6 Parameters

The RMIT-GP package [15] was modified to suit the problem. Clustering was performed using Cluster 3.0 [16]. Default genetic programming parameters for the RMIT-GP package were used, namely a mutation rate of 0.20, a crossover rate of 0.78 and an elitism rate of 0.02. Each run consisted of a population of 300 individuals evolved for 50 generations. The first generation of programs was generated randomly.

5 Experiments

We have conducted two experimental tests of the approach:

1. Learning set: 13 textures from the Brodatz database, train-and-test sets: 13 different textures from the Brodatz data base. These 13 textures are the same as the ones used in Wagner[17] to compare 18 human derived texture feature extraction methods. We have used the same methodology and the same training and test sets so that our results are directly comparable. There were 832 images (64 images per texture) for training and 1664 (128 images per texture) for testing. The test accuracy was 74.6%.

2. Learning set: 13 textures from the Brodatz database,

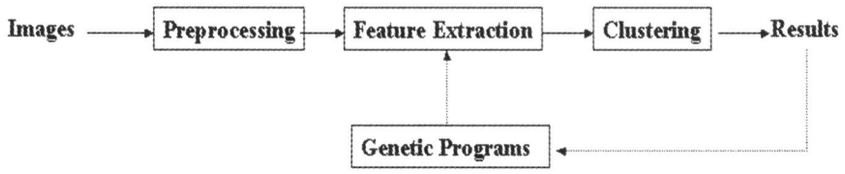


Figure 8: Feature Extraction Discovery Using Genetic Programming

train-and-test sets 15 textures from the Vistex data base. These 15 textures are the same as the ones used in Wagner[17] to compare 18 human derived texture feature extraction methods. We have used the same methodology and the same training and test sets so that our results are directly comparable. The Vistex data set has 480 images (32 images per texture) for training and 960 (64 images per texture) for testing. The accuracy obtained was 66.2%.

6 Analysis of An Example Program

Using a similar approach to that described in section 2, we will analyse an evolved program and see how it works for the more complex images of Bark, Brick and Fabric in the Vistex dataset. Recall from the example in section 2 that it is highly desirable for a feature extraction program to deliver outputs that are very similar to each other for the same texture and widely separated outputs for different textures. The simplified version of the program is P027 - P079. Binary images at grey level 98 from the training and testing set are shown in figure 9. The inputs and outputs are shown in table 2. The outputs of the program for brick train/test are -249/-258 which are quite close to each other, and well separated from the outputs for bark train/test which are 377/282, and the outputs for fabric which are 861/891.

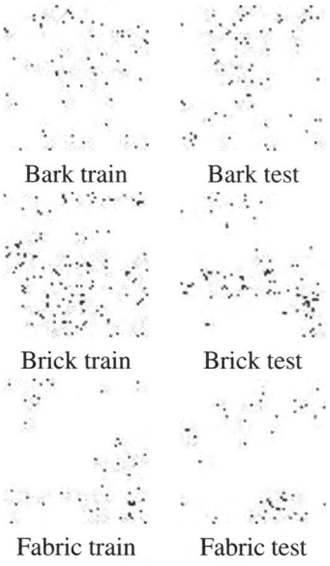


Figure 9: Bark, Brick and Fabric binary images at grey level 98 from training and testing sets

Table 2: Inputs and output of an evolved program for Bark, Brick and Fabric texture

	Bark		Brick		Fabric	
Pixel No.	Train	Test	Train	Test	Train	Test
P027	377	282	313	598	861	899
P079	0	0	664	856	0	0
Output	377	282	-249	-258	861	899

7 Results

Table 3 compares the performance of GP features against 18 human derived texture features. The last 3 lines show the results using genetic programming. "GP Spatials" are the results from the method presented in this paper. "GP Histograms" are results from [14] and "GP histograms + spatials" are the results from combining the features from "GP Spatials" and "GP Histograms".

On a 13 class problem it is possible to achieve an accuracy of 1 in 13 or only 8% by guessing, thus this result is a significant achievement. As can be seen from table 3, it has better accuracy than 4 human derived methods on the Brodatz problem and 5 on the Vistex problem.

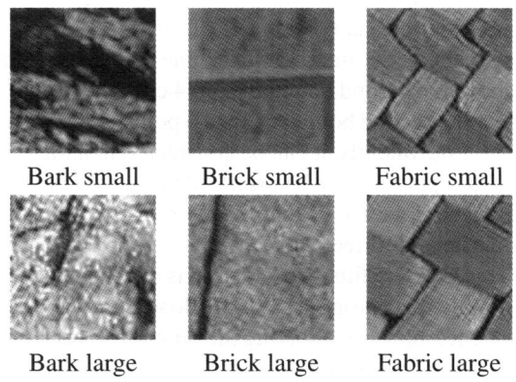


Figure 10: Bark, Brick and Fabric images at different scales

It is interesting to note that using spatial features is not as accurate as using histogram features of our previous work [14] (3rd last line of table 3). This is contrary to our expectations as the histogram features do not use any spatial information, a key aspect of texture. We believe that the reason for this is that the repeating patterns are not regular enough in some of the textures used. Rather than using all 256 grey levels for evolving features it may be better to quantize the

learning set to 64 or 32 grey levels and evolve only 64 or 32 feature extraction programs.

The last line of table 3 shows the results of combining the histogram features from [14] and the spatial features described in this paper. On the Brodatz problem, the addition of the spatial features results in a small increase in accuracy. However on the Vistex problem, there is a significant decrease. The combined approach has not worked as well for the Vistex images due to different spatial arrangements at the different scales mentioned in section 3, whereas the ratio of pixels at different scales are preserved at different scales for the histogram approach. See figure 10 for sample images at different scales.

Table 3: Performance of Various Feature Extraction Algorithms All results, except for the last 3 are from [17].

Feature Set	Brodatz	Vistex
Unser	92.6%	81.4%
Galloway	84.7%	70.4%
Laine	92.4%	75.6%
Local features	61.1%	47.1%
Fractal(1)	62.6%	54.5%
Fractal(2)	66.5%	48.5%
Laws	89.7%	79.8%
Fourier coeff.	92.7%	80.1%
Chen	93.1%	84.5%
Sun & Wee	63.9%	58.4%
Pikaz & Averbuch	79.4%	74.4%
Gabor	92.2%	75.4%
Markov	83.1%	69.6%
Dapeng	85.8%	74.6%
Amadasun	83.4%	65.6%
Mao & Jain	86.3%	73.0%
Amelung	93.0%	82.1%
Haralick	86.1%	75.5%
GP histograms	*81.5%*	*74.8%*
GP spatials	**74.6%**	**66.2%**
GP histograms + spatials	**83.3%**	**68.4%**

8 Conclusions and Future work

Our aim in this paper was to use genetic programming to evolve texture feature extraction programs that would be useful for a wide range of texture classification problems. Our evolved features have achieved good accuracy on a number of problems, but more work needs to be done to establish generality.

We have addressed the problem of how to capture spatial information in a texture for use in genetic programs for texture feature extraction. This is done by generating binary images for each grey level and encoding the positions of the black pixels as the terminals. A separate feature extraction program is evolved for each grey level. The programs capture the spatial differences between the textures.

We have shown how genetic programming can be configured to evolve texture feature extraction programs. A learning set of textures is needed, together with the spatial encoding and a fitness function based on measuring the tightness of clusters.

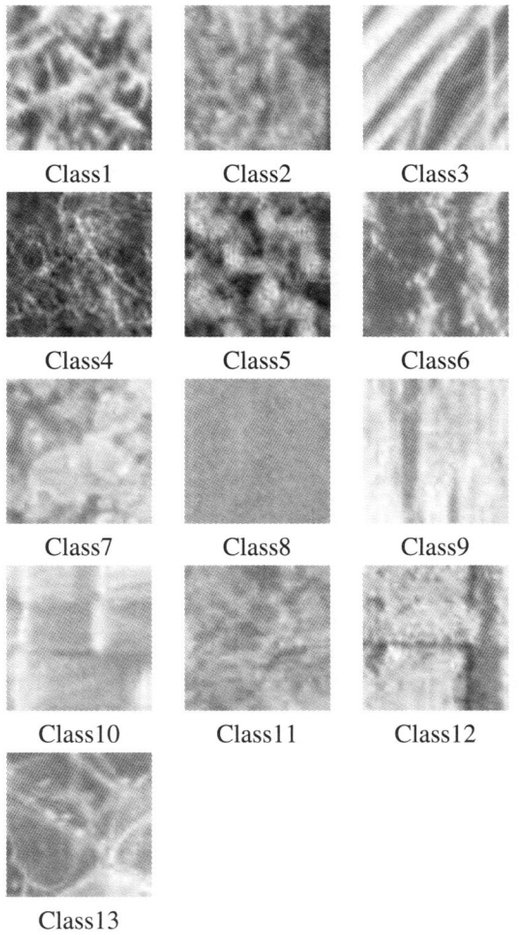


Figure 11: Brodatz Texture Images

While we have yet to test the approach on a wide range of texture classification problems, the results on a benchmark set of problems, in which 18 human derived methods are compared, are very encouraging. Our method is better than 4 and 5 human derived methods for the Brodatz and Vistex textures used.

Further work on the generality of the approach is needed. In particular on the choice of the learning set, and on identifying the texture regularities in the evolved programs with a view to understanding why they work well on textures not in the learning set.

Bibliography

[1] Tuceryan M. and Jain A.K., "Texture Analysis" in Handbook of Pattern Recognition and Image processing, World Scientific, 1993, Chapter 2, 235-276

[2] Koza J.R., Keane M.A., Streeter M.J., Mydlowec W., Yu J., Lanza G., "Genetic Programming IV Routine Human-Competitive Machine Intelligence" Kluwer, 2003, 1-10

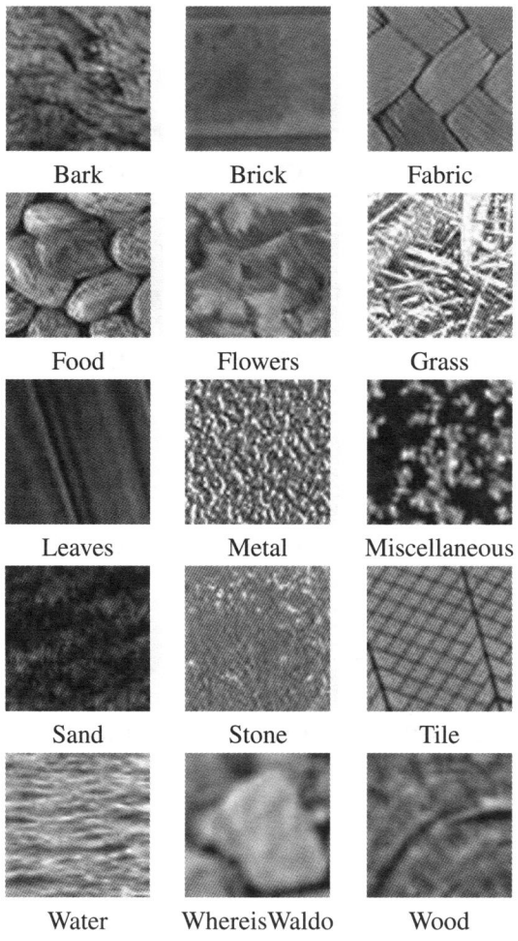


Figure 12: Vistex Texture Images

[3] Rignot E., Kwok R., "Extraction of Textural Features in SAR images: Statistical Model and Sensitivity" in Proceedings of International Geoscience and Remote Sensing Symposium, Washing DC, 1990, 1979-1982

[4] Jain A.K., Farrokhnia F., Alman D.H., "Texture Analysis of Automotive Finishes" in Proceedings of SME Machine Vision Applications Conference, Detroit, 1990, 1-16

[5] Chen C.C., Daponte J.S., Fox M.D., "Fractal Feature Analysis and Classification in Medical Imaging" IEEE Transactions on Medical Imaging, 1989, 8, 133-142

[6] Jain A.K., Bhattacharjee S.K., Chen Y., "On Texture in Document Images" in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Champaign, Il, 1992

[7] Ross B.J., Fueten F. and Yashkir D.Y., "Automatic Mineral Identification using Genetic Programming", in Technical Report CS-99-04, Brock University, December 1999

[8] Lin Y. and Bhanu B., "Learning Features for Object Recognition", GEC2003, LNCS2724, 2003, 2227-2239

[9] Daida J., Hommes J., Bersano-Begey T., Ross S. and Vesecky J., "Algorithm Discovery Using the Genetic Programming Paradigm: Extracting Low-Contrast Curvilinear Features from SAR Images of Arctic Ice", "Advances in Genetic Programming 2",1996,MIT Press, 417-442

[10] Howard D. and Roberts S., "Evolving object detectors for infrared imagery: a comparison of texture analysis against simple statistics", "Evolutionary Algorithms in Engineering and Computer Science", 1999, 79-86

[11] Song S., Ciesielski V. and Williams H, "Texture Classifiers Generated by Genetic Programming", "Proceedings of the 2002 Congress on Evolutionary Computation CEC2002",2002,243–248

[12] Koeppen M. and Liu X., "Texture Detection by Genetic Programming", "Proceedings of the 2001 Congress on Evolutionary Computation CEC2001", 2001, 867–872

[13] Kueblbeck C., "Optimized configuration of systems for texture analysis","Proceedings of SPIE Volume 3966, Conference 3966A: Machine Vision Applications in Industrial Inspection VIII and Conference 3966B: Surface Characterization for Computer Disks, Wafers, and Flat Panel Displays II",2000,123-133

[14] Lam B. and Ciesielski V., "Discovery of Human-Competitive Image Texture Feature Extraction Programs Using Genetic Programming", "Genetic and Evolutionary Computation – GECCO-2004, Part II",2004, 1114–1125

[15] Mawhinney D., RMIT-GP version 1.3.1, the RMIT University, 2002, http://goanna.cs.rmit.edu.au/ vc/research.html

[16] De Hoon M., Human Genome Center, University of Tokyo, http://bonsai.ims.u-tokyo.ac.jp/ mdehoon/software/cluster/software.htm

[17] Wagner T. , "Texture Analysis" in Handbook of Computer Vision and Applications, 1999, Volume 2, Chapter 12, 276-308