Brock University

Department of Computer Science

Feature Extraction Languages and Visual Pattern Recognition

Mehran Maghoumi and Brian J. Ross

Technical Report # CS-14-03
January 2014

Brock University
Department of Computer Science
St. Catharines, Ontario Canada L2S 3A1
www.cosc.brocku.ca

Feature Extraction Languages and Visual Pattern Recognition

Mehran Maghoumi
mm12tm@brocku.ca

Brian J. Ross
bross@brocku.ca

ABSTRACT

Visual pattern recognition and classification is a challenging computer vision problem. Genetic programming has been applied towards automatic visual pattern recognition. An important factor in evolving effective classifiers is the suitability of the GP language for defining expressions for feature extraction and classification. This research presents a comparative study of a variety of GP languages suitable for classification. Four different languages are examined, which use different selections of image processing operators. One of the languages does block classification, which means that an entire region of pixels is classified simultaneously. The other languages are pixel classifiers, which determine classification for a single pixel. Pixel classifiers are more common in the GP-vision literature. We tested the languages on different instances of Brodatz textures, as well as aerial and camera images. Our results show that the most effective languages are pixel-based ones with spatial operators. However, as is to be expected, the nature of the image will naturally determine the effectiveness of the language used.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Experimentation

Keywords

genetic programming, feature extraction, pattern classification, texture, computer vision

1. INTRODUCTION

Pattern recognition and classification is a challenging computer vision problem. In basic terms, visual pattern classification involves the automatic identification of some image feature of interest. For example, one may want a computer program to automatically differentiate given different bitmap images of textures (wood grain, cloth, fur,...). The automatic recognition of more complex patterns is also possible, such as fingerprints, faces, and military targets in satellite images.

Genetic programming (GP) has been used in computer vision classification problems [4]. GP has been shown to be effective in a variety of image analysis tasks such as texture classification [16, 13] and image and texture segmentation [10, 14, 15]. GP has also been applied to other complex computer vision problems such as target and object detection [6, 5, 17], face detection [18], mineral identification [12], and medical image analysis [11]. More recently, GP has been applied to real-time computer vision problems such as adaptive [2] and automated [3] object recognition.

Much of the aforementioned GP vision research use pixel-based classifiers. These classifiers involve GP expressions that examine features of a subject image centred around a pixel of interest, and determine a classification result for that pixel. To process an entire image, the GP tree is executed consecutively on all the pixels in an image. Although this has the potential of very precise pixel-level classification, it can be computationally expensive – and especially so for large GP expressions and high resolution images.

Song *et al.*'s work in GP vision uses a different approach – block (or region) classification [13, 14, 15, 16]. Here, the GP tree also accesses features of a region of an image, for example, a 32×32 square block. After one single execution of the classifier expression, a classification is made for the entire region of the image. They reported very good results with block classification. Furthermore, block classifiers are much more efficient than pixel classifiers – in fact, 1024 times more efficient using 32×32-sized blocks.

The motivation of this paper is to more closely examine the effect of GP-based feature extraction languages on visual pattern recognition. In particular, we compare performance of block- and pixel-based classifiers on a different pattern recognition problems, for example, Brodatz textures[1], as used in [13, 14, 15, 16], as well as some satellite and camera images.

Although it is clear that block classifiers are more efficient than pixel classifiers, we are primarily interested here in comparing performance accuracy between the different pattern classification languages. Our curiosity is motivated by the fact that block classifiers such as in [13, 14, 15, 16] examine a very sparse selection of data compared to pixel classifiers typically found in the GP vison literature.

The paper is organized as follows. The pattern classification problems to be examined are introduced in Section 2. Four classification languages are defined in Section 3, and other experimental details are given in Section 4. Results of our experiments are presented in Section 5, and further discussed in Section 6. Section 7 summarizes the paper, and discusses directions for future research.

.

2. PATTERN CLASSIFICATION PROBLEMS

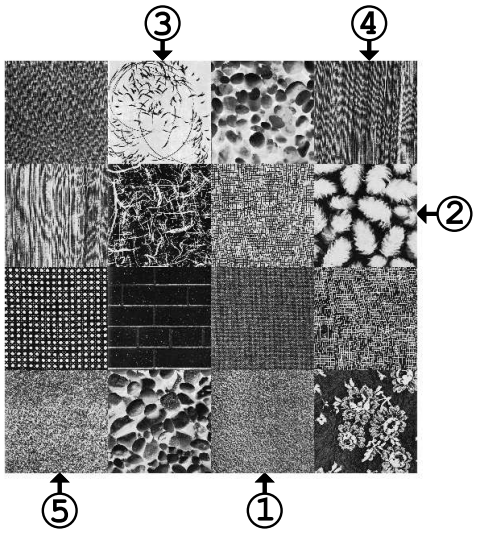


Figure 1: Brodatz textures. (Full image is 512×512 pixels, and each texture area is 128×128.)

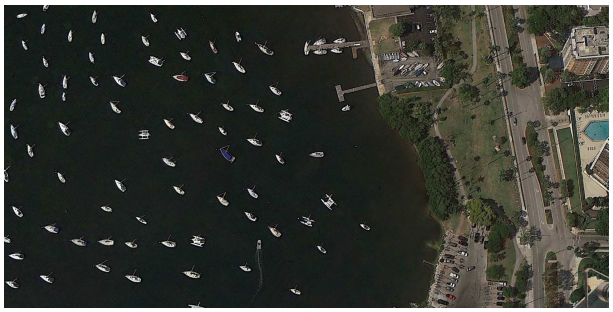


Figure 2: Aerial image of boats. (1692×843 pixels.)

The main vision problem of concern is the recognition of textures taken from the Brodatz texture data set[1][1], which were used previously in [13, 14, 15, 16]. Figure 1 shows the textures we have used. The 5 numbered textures are used separately as positive examples. For example, when texture 1 is a positive example, the remaining 15 textures are used as negative cases. The textures were selected to introduce various levels of difficulty. Some textures such as 1 and 5, have very similar patterns while other textures such as 3, are easily distinguished from other textures in the set.

We also examine two other images, to give potential insights of the pattern recognition languages on other kinds of vision problems. We examine an aerial image of boats near a port obtained from the satellite view of Google Maps[2] (Figure 2). We wish to detect boats on water. We also use a photo of a group of people (Figure 3), in which we wish to identify human faces from the rest of the image. Ground truth images were manually made to indicate the areas of the images with boats and faces – in other words, image areas to be considered as "positive" or true.

[1]http://www.ux.uis.no/∼tranden/brodatz.html
[2]http://maps.google.com/

Figure 3: Group photo. (1280×720 pixels.)

3. CLASSIFICATION LANGUAGES

A goal of this paper is to compare two different approaches to GP-evolved pattern classifiers – pixel classifiers and block (or region) classifiers. A pixel classifier makes a classification decision for a single pixel of interest, normally by examining spatial information in the surrounding image region of that pixel. A block classifier makes a decision for all the pixels within a region, based on information in that region. Pixel-based classifiers are more common in the GP literature (see Section 1), while block-based classifiers are used by Song *et al*. [13, 14, 15, 16].

In the following, we will describe both of these methodologies in detail. Since they are highly dependent upon the GP language used to define the classifier, the four GP languages studied (one block language, and three pixel languages) are also described.

3.1 Block classifier language

A block classifier language was presented by Song *et al*. [13, 16], and has been used in other vision applications [14, 15]. We implemented a block language resembling Song *et al*.'s original language. To perform block classification, three images are provided to the system: a target image for positive classification, the target for negative classification, and a test image. Training instances are created by randomly sampling 400 sub-image *blocks* each from the positive and negative target images. Our blocks are of size *32×32* pixels. The blocks contain various grey-scale (and possibly colour) information about the positive and negative target images, and comprise a feature vector for the GP system. Using the feature vector, an evolved GP tree composed of various mathematical and decision making functions can extract classification information from each block. This results in a binary classifier which, when applied to an image region, makes a classification decision for that entire region. The goal for evolution is to evolve a classifier that correctly classifies positive and negative image instances, on a block-by-block basis.

The block classification language is given in Table 1, and uses functions and terminals standard in the literature. Att[x] represents the x^{th} pixel in the block (modulo total pixels in the block), and returns the RGB or grey-scale value. Strongly typed GP is used for this and the other languages [8]. The language shown has Boolean and double data types. The root of each GP tree is double. Computed values greater than or equal to zero are interpreted as *true*, while negative are *false*.

When a block classifier makes a decision about a 32×32 region, all 1024 pixels in that region are assigned that classification. Therefore, for the majority of the pixels in an image, there are 1024 potential ways for assigning a classification to each pixel, depending on the placement of the 32×32 block overlaying that pixel. This

Table 1: Block classification language. (D=Double, B=Boolean)

Name	Return type	Argument type	Description
Add	D	D	addition
Sub	D	D	subtraction
Mul	D	D	multiplication
Div	D	D	protected division
If	D	B, D, D	if a true then b else c
>=	B	D, D	true if $a \geq b$
=<	B	D, D	true if $a \leq b$
Between	B	D, D, D	true if $b < a < c$
Random	D	-	random constant, -1 $\leq c \leq$ 1
Att[x]	D	I	Value of attribute

poses some difficulty in comparing block classifiers with pixel classifiers, since the odds are good that there is at least a few block placements which give contradictory classifications for any given pixel. Therefore, in order to make a more meaningful comparison between block and pixel classifiers, we use the following approach. During testing, block classifiers are applied to images by placing the block area over every possible 32×32 region of an image (edges are not crossed, and so pixels close to edges have fewer block overlays). We then tally the number of times each pixel was identified as "true" within a tested block, scaled by the number of times that pixel was processed in total. This results in a percentage value that each pixel was classified as true. Should a pixel be classified as true the majority of time (ie. threshold of 50%), then it is considered to be true with respect to performance measurements.

3.2 Pixel classifier languages

Table 3: Pixel language part 2. (I=integer, D=double)

Name	Return type	Argument type	Description
Add	I/D	I/D	addition
Sub	I/D	I/D	subtraction
Mul	I/D	I/D	multiplication
Div	I/D	I/D	protected division
Neg	D	D	negation
Exp	D	D	e raised to the operand
IfGT	D	D,D,D,D	if $a > b$ then c else d
Max	D	D,D	maximum
Min	D	D,D	minimum
Sin	D	D	sine
Cos	D	D	cosine

Pixel-based classification works as follows. For a given data set, two images are provided to the GP system – an image to process, and a ground truth image. The ground truth image is marked to show the positive region(s) to be identified in the image. Using the ground truth, the system randomly samples 512 positive pixels and 1024 negative pixels from the input image. These *centre pixels* are then used for creating training instances. Using the coordinates of the centre pixels, a block of n×n pixels is formed around these pixels in a way that the centre pixels coincides with the coordinates ($\lfloor n/2 \rfloor$, $\lfloor n/2 \rfloor$). We refer to these blocks as *grids*. For each pixel in the grid, spatial filter values (average and standard deviation) are calculated and stored for later access by GP expressions. To speed up processing, we compute these values using NVIDIA CUDA [9].

During training, a binary classifier is evolved using positive and negative training instances. The raw pixel values, as well as the spatial filter values, form the feature matrix for the system. GP uses this matrix in conjunction with mathematical and decision making functions to evolve a classifier. Integer offsets can be used to extract features in the vicinity of the centre pixel. The decision made by the classifier is applied to the centre pixel. During testing, the classifier processes every pixel of the image. This contrasts to the block classifier, which assigns the classification to all the pixels in the region.

Table 4: Summary of the language variations

Name	Spatial operations	Offsets	Block processing
Complete	×	×	
No Offset	×		
Raw Features		×	
Block Processing		×	×

We define 3 pixel classification languages, which will use functions and terminals selected from Tables 2 and 3. The languages use 3 data types: double, integer, and channel. Terminals (see Table 2) include ephemeral random constants for integers and doubles, channel index, and random terminals (every access generates a new random value). Pixel values (RGB and/or grey-scale) are accessible, either directly for a centre pixel, or for a specified integer offset within the 32×32 block. The channel argument c specifies the particular channel (R, G, B, grey-scale) to retrieve. Grey-scale images have R, G, and B removed. Spatial data (average, standard deviation) can also be read for centre pixels and offsets near them, again for the specified channel. The area values (15, 17, 19) were determined by experimentation, as earlier attempts using smaller areas were not beneficial. Table 3 shows the remaining functions, which are standard in the literature. There are integer and double versions of the basic arithmetic operators.

The four languages used for the experiments are summarized in Table 4. *Spatial operations* refer to the average and standard deviation functions, while *offsets* include the colour and spatial operators that use (i,j) offsets. The first two languages (Complete, No Offset) thus use spatial operators, while the others do not. Raw Features is essentially the Block Processing language with extended mathematical operators, but to be used in a pixel-classification manner.

4. EXPERIMENT SETUP

Table 5: Run Parameters

Parameter	Value
Population size	1024
Generation size	50
Crossover rate	90%
Mutation rate	10%
Selection method	Tournament selection
Tournament size	4
Elites	2
Number of Runs	20
Pixel block size	32×32

In all runs, the static range selection method [7] was used. A GP expression is evaluated for a pixel (block), and if the output is ≥ 0, it is considered a positive classification. This is compared to the ground truth image. The number of true positives and true negatives are then used to calculate the fitness value of the individual in

Table 2: Pixel language part 1. (I=integer, D=double, c=channel)

Name	Return type	Argument type	Description
c	c	-	channel index (0,1,2,3)
ERC	D	-	ephemeral random constant in the range [0, 1]
Random	I	-	random integer in the range [0, 31]
GridERC	I	-	ephemeral random constant in the range [0, 31]
Input colour	D	c	channel value of the selected pixel
$Avg_{k=15,17,19}$	D	c	average of $k \times k$ area
$Stdev_{k=15,17,19}$	D	c	standard deviation of $k \times k$ area
Input colour	D	c, I, I	channel value c at (i,j) offset
$GAvg_{k=15,17,19}$	D	c, I, I	average of $k \times k$ area of channel c, offset (i,j)
$GStdev_{k=15,17,19}$	D	c, I, I	standard deviation of $k \times k$ area of channel c, offset (i,j)

question:

$$fitness = \frac{TP + TN}{TOTAL} \times 100 \quad (1)$$

where TP is the number of true positives, TN is true negatives, and TOTAL is the total number of cases.

Table 5 summarizes the GP parameters used.

5. RESULTS

Table 6 shows the final score results of the experiments. Most scores are averaged over 20 runs, except for *best total*, which is the overall image score of the single top performing classifier found. Testing scores are the true positive and true negative scores for the image, with training pixels removed. Total are the true positive, true negative, and all pixels combined score for the entire image (including training pixels). We have tagged the best overall performing languages (within a statistical significance measure of 95%) for each image studied. Statistical significance was measured with an unpaired t-test with unequal variance.

Firstly, training and testing scores for true positives/negatives are closely matching in all experiments, and so over-training is unlikely to be occurring.

There is some variability apparent in performance for different languages and images. For example, for Texture 1, the Raw Features language is weaker at positive identification than the other languages. The Block Processing language, however, is weak in negative identification. Overall the spatial languages do better on Texture 1, with the No Offset language being the best performer.

Texture 4 was the most challenging image for the languages. The No Offset language was again the best performer. Surprisingly, most solutions from the Complete language were weak, mostly due to poor positive recognition performance (low true positive scores).

Summarizing the scores in Table 6, the No Offsets spatial language was the overall top performing language, having the best (statistically significant) average solution performance for 5 images. This is followed by the Complete spatial language (2 images), and the Block Processing language (2 images). Together, the spatial languages (Complete, No Offset) are the overall top performers, meaning that spatial operators are useful for many images studied. Curiously, even though the Complete language is a super-set of the No Offset language, it was not as good a performer in many instances. This contradicts conventional wisdom that GP evolution will naturally select the best language operators for a problem at hand. In our experience, the more complex language was not refined by evolution. This may be due to the larger language defining too complex a search space for the image at hand. On the other hand, the Raw Features and Block Processing languages do not use spatial operators. It is clear that the block processing strategy we used with the Block Processor is advantageous for that language, as it is the main technical difference between it and the Raw Features language, which is the poorest performing language of the four studied.

Figure 4 show some texture image results of the experiments. Images (a-d) are the best solution image results for Texture 1 recognition, and (e-h) for Texture 2. The overall scores for these images are found in the "best total" rows in Table 6. Green indicates positive identification. Therefore, the high density of green in the Texture 1 area (see Figure 1) that is apparent in images (a-d) corresponds to the high true positive scores. Green in other texture areas denotes false positives. Hence, one can see where the Complete, No Offset, and Block Processing languages were tricked by the texture in the bottom-left corner. Similarly, the Raw Features language was fairly liberal in (c) when identifying positive instances, resulting in a low overall score of 65.6%.

Texture 2 is one of the more difficult textures to recognize. The results in (e-h) show that it is usually recognized, but more mistakes (false positives and negatives) arise in (e,f,g). The Block Processing results in (h) have a high degree of false positive. A good result of the Block Processing language on Texture 3 is shown in (i).

Also note that inter-texture boundaries can be difficult for some languages. This is because boundaries represent complex mixes of different textures. The spatial operators are free to overlay across these boundaries, which can complicate recognition of positive regions. We consider boundary artifacts to be acceptable noise.

Figure 5 show sample results for the aerial boat image. The No Offset result in (b) is very close to the ground truth (a), and has an 88.05% image score. False positive are mostly found in the ground clutter on the right-side. For comparison, a Block Processing result is shown in (c). The details in (d) show the primary advantage of pixel-based classification over block processing — the ability to do precise classification.

Figure 6 show some results with the group photo. The faces are identified, but with quite a lot of false positive results on arms and background. The block result shown is much worse. In hindsight, this is a challenging image to analyze, as the faces are very difficult to ascertain without higher-level image processing. Although the group photo's results are worse than the textures and boats, they are useful in that they highlight some natural limitations of the languages studied for difficult images.

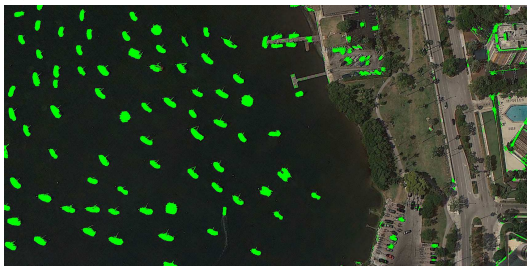
Figure 5 shows the fitness performance for the Texture 1 runs. The top 4 curves are for the spatial languages, which show superior fitness to the non-spatial languages.

6. DISCUSSION

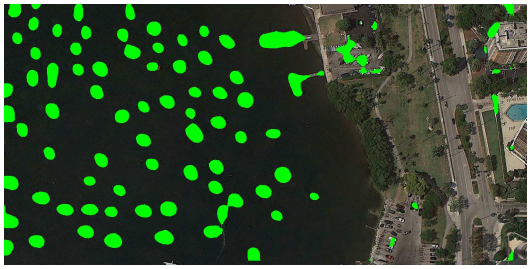
Table 6: Experiment results. All scores are %. "test +/-" scores are for testing, "total +/-" is entire image, (including training pixels), and total is overall average image score. All scores are averaged over 20 runs, except for "best total", which is image score from single best classifier found. Top performing overall languages (within statistical significance of 95%) tagged * and highlighted in bold.

		Language			
Image	**Scores**	**Complete**	**No Offset**	**Raw Feat.**	**Block Proc.**
Texture 1	test +	79.40	89.91	39.69	95.73
	test -	93.67	93.73	83.75	35.33
	total +	79.44	89.96	39.78	97.94
	total -	93.67	93.73	83.75	35.41
	total	86.55	***91.85**	61.76	66.68
	best total	94.17	93.51	65.60	81.34
Texture 2	test +	75.55	58.20	25.76	97.40
	test -	85.74	90.20	90.64	60.68
	total +	75.62	58.28	25.78	99.49
	total -	85.75	90.21	90.65	60.78
	total	***80.68**	74.24	58.21	***80.13**
	best total	89.78	82.68	62.21	85.32
Texture 3	test +	96.76	96.39	73.68	94.66
	test -	97.68	97.64	87.78	98.05
	total +	96.77	96.41	73.70	97.12
	total -	97.69	97.64	87.78	98.21
	total	97.23	97.03	80.73	***97.67**
	best total	98.19	98.25	81.41	98.27
Texture 4	test +	27.33	67.51	2.41	94.23
	test -	91.94	88.70	98.23	41.95
	total +	27.37	67.55	2.42	96.46
	total -	91.94	88.70	98.23	42.03
	total	59.66	***78.12**	50.33	69.24
	best total	84.01	86.05	51.65	70.41
Texture 5	test +	81.85	87.37	20.09	89.79
	test -	86.32	89.41	90.28	57.14
	total +	81.90	87.41	20.15	92.01
	total -	86.33	89.41	90.28	57.24
	total	84.11	***88.41**	55.21	74.63
	best total	91.06	90.71	62.78	80.44
Boats	test +	94.53	94.74	77.25	95.89
	test -	96.21	96.45	93.35	67.11
	total +	94.56	94.79	77.31	97.76
	total -	96.21	96.45	93.35	67.14
	total	***95.38**	***95.62**	85.33	82.45
	best total	96.31	96.83	88.05	93.59
Face	test +	90.94	95.28	87.90	94.97
	test -	92.65	94.26	89.86	64.69
	total +	91.01	95.37	87.99	99.29
	total -	92.65	94.26	89.86	64.72
	total	91.83	***94.81**	88.92	82.00
	best total	95.71	96.96	90.37	94.71

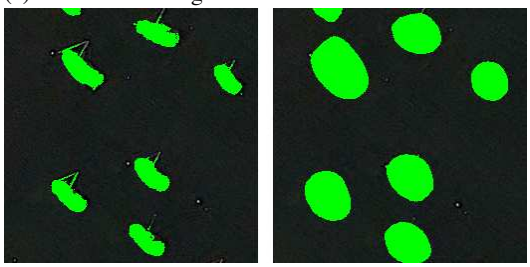
(a) Ground truth.



(b) No Offset.



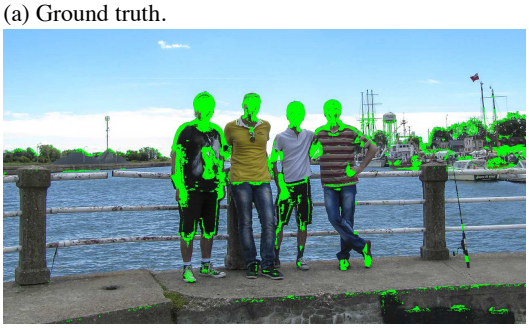
(c) Block Processing.



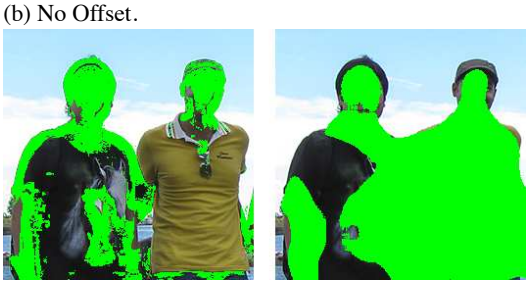
(d) Details: (left) No Offset, (right) Block Proc.

Figure 5: Boat output images

(a) Ground truth.



(b) No Offset.



(c) Details: (left) No Offset, (right) Block Proc.

Figure 6: Face output images

Our results show that pixel-classification languages with spatial operators are preferable to those without spatial operators, and to block-classification languages. As described in Section 3.1, our block processing strategy used to analyze the results in Table 6 is not the manner in which other papers used block languages [13, 14, 15, 16]. Our initial analysis applied the block language on a block-by-block bases on images, similar to [13]. The resulting test scores were quite low, because block-based classification is hit-or-miss, and mistakes are costly. By using exhaustive block overlays and thresholding, we improved the block language performance.

Examining research in [13, 16, 14, 15], we note that the GP expressions used for block processing were apparently not applied in a random-sampled manner to images. Rather, fixed coordinate positions for block overlays were used on training images, which also apparently coincide to the overlays used during testing. This means that the same relatively small set of image samples were seen during training and testing, and the resulting testing performance can be respectable. Computational performance in wall-clock speed is also advantaged by the ability to classify large regions of images at once.

Our experience is that using a block classifier with random sampling of images during training and testing is more challenging for block classification. Since the simple block language samples only a relatively sparse number of pixel points in an image region, it is difficult to learn pattern concepts for complex images when too sparse a set of training points are used. Spatial languages overcome this due to their ability to extract pertinent features over an image region, and in our case, using average and standard deviation calculations (which are akin to blur and edge filters). Furthermore, because a pixel-based spatial language can error for one pixel rather than an entire region, errors are far less costly.

7. CONCLUSION

Our results show that spatial pixel classifiers are the best performers for the problems we studied. The rich set of spatial operators used means that they are capable of sophisticated feature extraction compared to other languages studied. We found that the block classifier was competitive with the pixel classifiers in the test environment undertaken. However, it should be realized that block classifiers are not intended to be run repeatedly on images in the way we did during testing. In this sense, we gave the block lan-

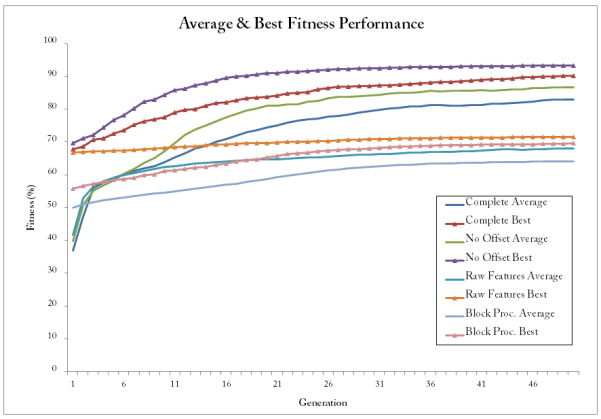


Figure 7: Fitness performance graph for Texture 1 runs (average 20 runs).

guage some advantages, since the classification result of a pixel was determined after typically 1024 GP tree executions (compared to the usual single execution [13]). Nevertheless, our test results show that the block classifier language studied has respectable capabilities, considering the simplicity of the language used.

There are many directions for future investigations. The GP vision literature (see Section 1) shows that much more complex GP languages for pattern classification are possible. The language definitions we used could be made considerably more advanced, by considering sophisticated image processing primitives. The degree to which complex languages are required, however, depends upon the application. Song *et al*.'s work shows that a very rudimentary language is capable of texture recognition, and there is no reason to believe that all problems require complex primitives. Future work should also consider alternate kinds of vision applications. New problems will undoubtedly require the use of appropriate pattern classification languages.

Another direction of research is the application of GP-evolved classification languages towards image segmentation. Our "window sweeping" of classifiers on the aerial boat image produced interesting results, and especially so with the block classifier language. Although the block thresholding we used is slow compared to the normal block classifier approach, it is more accurate than standard block classification, and has the same efficiency as pixel classifiers. We intend to exploring this new application further, and are considering implementation with NVIDIA CUDA to speed up processing [9].

8. REFERENCES

[1] P. Brodatz. *Textures: a photographic album for artists and designers*, volume 66. Dover New York, 1966.

[2] M. Ebner. A real-time evolutionary object recognition system. In *Proceedings of the 12th European Conference on Genetic Programming*, EuroGP '09, pages 268–279, Berlin, Heidelberg, 2009. Springer-Verlag.

[3] M. Ebner. Towards automated learning of object detectors. In *Proceedings of the 2010 international conference on Applications of Evolutionary Computation - Volume Part I*, EvoApplicatons'10, pages 231–240, Berlin, Heidelberg, 2010. Springer-Verlag.

[4] P. Espejo, S. Ventura, and F. Herrera. A survey on the application of genetic programming to classification. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(2):121–144, 2010.

[5] N. Harvey, S. Perkins, S. Brumby, J. Theiler, R. Porter, A. Cody Young, A. Varghese, J. Szymanski, and J. Bloch. Finding golf courses: The ultra high tech approach. In S. Cagnoni, editor, *Real-World Applications of Evolutionary Computing*, volume 1803 of *Lecture Notes in Computer Science*, pages 54–64. Springer Berlin Heidelberg, 2000.

[6] D. Howard, S. C. Roberts, and R. Brankin. Target detection in sar imagery by genetic programming. *Adv. Eng. Softw.*, 30(5):303–311, May 1999.

[7] T. Loveard and V. Ciesielski. Representing classification problems in genetic programming. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 2, pages 1070–1077 vol. 2, 2001.

[8] D. Montana. Strongly Typed Genetic Programming. *Evolutionary Computation*, 3(2):199–230, 1995.

[9] J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable parallel programming with cuda. *Queue*, 6(2):40–53, Mar. 2008.

[10] R. Poli. Genetic programming for feature detection and image segmentation. In T. Fogarty, editor, *Evolutionary Computing*, volume 1143 of *Lecture Notes in Computer Science*, pages 110–125. Springer Berlin Heidelberg, 1996.

[11] R. Poli. Genetic programming for image analysis. In *Proceedings of the First Annual Conference on Genetic Programming*, pages 363–368. MIT Press, 1996.

[12] B. Ross, A. Gualtieri, F. Fueten, and P. Budkewitsch. Hyperspectral Image Analysis Using Genetic Programming. *Applied Soft Computing*, 5(2):147–156, 2005.

[13] A. Song. *Texture Classification: a Genetic Programming Approach*. PhD thesis, RMIT University, April 2003.

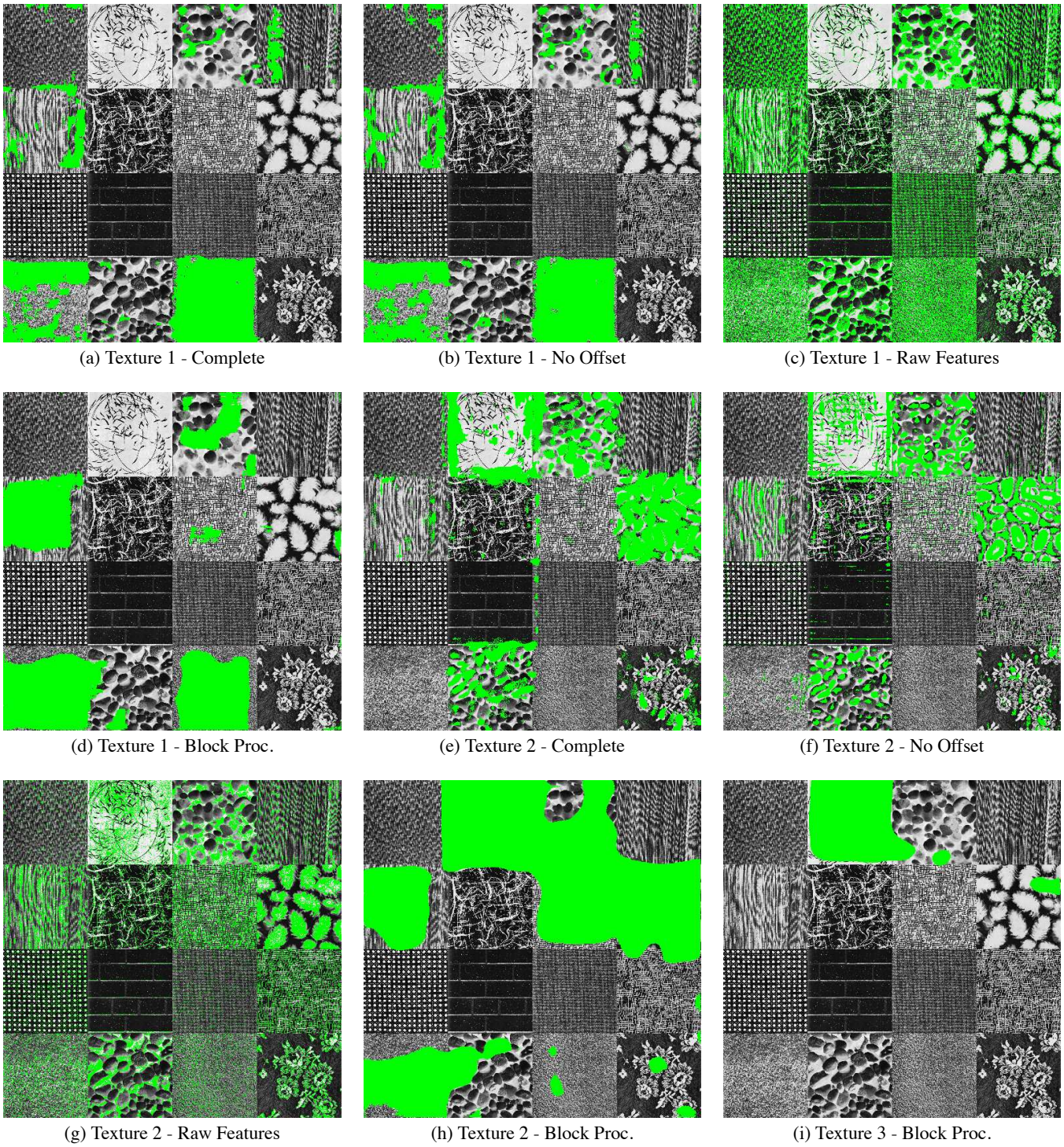
[14] A. Song and V. Ciesielski. Fast texture segmentation using genetic programming. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, volume 3, pages 2126–2133 Vol.3, 2003.

[15] A. Song and V. Ciesielski. Texture analysis by genetic programming. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 2092–2099 Vol.2, 2004.

[16] A. Song, T. Loveard, and V. Ciesielski. Towards genetic programming for texture classification. In M. Stumptner, D. Corbett, and M. Brooks, editors, *AI 2001: Advances in Artificial Intelligence*, volume 2256 of *Lecture Notes in Computer Science*, pages 461–472. Springer Berlin Heidelberg, 2001.

[17] W. A. Tackett. Genetic programming for feature discovery and image discrimination. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 303–311, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[18] J. F. Winkeler and B. Manjunath. Genetic programming for object detection. In *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 330–335. Morgan Kaufmann, 1997.



(a) Texture 1 - Complete
(b) Texture 1 - No Offset
(c) Texture 1 - Raw Features
(d) Texture 1 - Block Proc.
(e) Texture 2 - Complete
(f) Texture 2 - No Offset
(g) Texture 2 - Raw Features
(h) Texture 2 - Block Proc.
(i) Texture 3 - Block Proc.

Figure 4: Texture output images