The University of Saskatchewan
Department of Computer Science

Technical Report #2004-04

Texture Analysis of Compressed Images

Mark G. Eramian [1]

Department of Computer Science, 57 Campus Drive, The University of Saskatchewan, Saskatoon, Saskatchewan, Canada, S7N 5A7

Abstract

We introduce a new method of image texture analysis that is based on techniques that compress images in the form of weighted finite automata (WFA). We show that the WFA representation of images contain information about an image that is useful in analysing local texture and define a simple texture analysis method based on WFA. This method is the first texture analysis method that uses weighted finite automata and demonstrates the potential of WFA for further developments in texture analysis and segmentation. We include experiments on sample images showing the results of our technique and discuss certain advantages that our method has over other types of methods.

Key words: texture analysis, texture segmentation, weighted automata

1 Introduction

Recently there has been considerable research on image compression techniques using weighted finite automata. Weighted finite automata were first introduced by Culik [1] and are very similar to the classes of automata presented by Santos [2]. The idea of encoding greyscale images as a weighted finite automaton (WFA) was introduced by Culik [1] and he showed that the automaton representation can be considerably smaller than the original bitmap resulting in compression rates rivaling those of JPEG compression [3]. Typically, images are encoded using a quadtree decomposition which generates a multi-resolution image. Each unique image subsquare formed by this decomposition is assigned to a state in a weighted finite automaton. Transitions

Email address: eramian@cs.usask.ca (Mark G. Eramian).

[1] This resarch was funded in part by the Natural Sciences and Engineering Research Council of Canada through grant number OPG0000243 (H. Jürgensen) and through institutional grants provided by the University of Saskatchewan.

exist between states when one state's corresponding subimage is a subquadrant of another's with a possible change in contrast given by the transition weight. In theory, the encoded image can be decompressed to a bitmap of the original image at any finite resolution $2^k \times 2^k$ by running all words of length k on the automaton. A variety of WFA methods for compression have also been studied by Lin [4], Katritzke [5], Litow and de Vel [6], and Hafner [7] who describes a variation that uses a binary tree decomposition. We propose that the WFA representation is well-suited to image texture analysis due to the multiresolution information contained therein. In this paper we outline a method of texture analysis that takes advantage of this property of the WFA encoding.

In their survey of texture analysis, Tuceryan and Jain [8] suggest that the various methods of texture analysis can be used to solve four basic types of problems, namely texture segmentation, texture classification, texture synthesis, and shape from texture. The model for texture analysis we present in this paper is suited for the segmentation and classification problems. Our method however, does not assume any knowledge of the image to be analyzed so it does not fit perfectly into the category of classificiation methods which attempt to match an image or areas of an image with known texture classes. Instead we will define broadly what we mean by "highly textured" versus "highly untextured" and classify areas of the image based on these relative definitions. Neither does our method truly attempt to perform a segmentation of the image into regions of homogeneous texture, although it has the potential to be used as part of a larger segmentation scheme. This is the first proposed texture analysis technique that uses weighted finite automata and demonstrates the potential of WFA for use in texture analysis.

Some excellent survey papers such as those by Van Gool et. al [9], Reed and Du Buf [10], Haralick [11] and more recently, Tuceryan and Jain [8] have attempted to classify the vast literature of texture analysis methods by grouping them into a handful of general categores. These, in general, are region-based methods, boundary-based methods, transform-domain methods, statistical methods and geometrical methods.

Early region-based methods, such as that used by Reed and Werman [12], gradually expand homogeneous texture regions from seed points, but this requires some *a-priori* knowledge of the number and locations of textures in the image in order to position the seed points. Newer "unseeded" region-growing methods do not require this knowledge as demonstrated by Hojjatoleslami and Kittler [13]. In general, the performance of these algorithms degrade when the boundaries between texture regions are not distinct.

Boundary-based or edge-based methods (e.g. Khotanzad and Chen [14]) try to find the edges of homogeneous textures and work best when these boundaries

are crisp and well-defined. Newer methods that combine boundary-based and region-based approaches, such as Paragios and Deriche [15] have also been successful.

Model-based methods also require *a-priori* knowledge of the number and types of textures in an image as they attempt to match regions of an image to pre-computed statistical models of expected textures in order to perform image segmentation. Gibbs random fields used by Derin and Elliot [16] and Markov random fields used by Krishnamachari and Chellappa [17] have been popular methods.

Transform-domain methods perform their analysis in some other domain rather than on the image pixels themselves. The most common transform domains are the Fourier and wavelet domains and examples of these can be found in Van Gool et. al [9] and Unser [18]. Fourier domain methods suffer from the disadvantage that standard Fourier spectra contain no local information, though some methods have overcome this to a certain extent, for example, Eramian et. al. [19]. Wavelet domain methods are popular and have the advantage of allowing for the use of any number of basic wavelet sets, but *a-priori* information is needed to select the most optimal of such sets. In these methods, the transform process itself can take a prohibitive amount of computer time depending on the application, sizes of images, and the need for speed, however, this issue is slowly being overcome by technology advances.

Statistical methods work by computing texture features based on the spatial distribution of grey values in an image. Haralick's grey level co-occurance matrices [20] are the most widely used of such texture features. These methods are not well-suited to segmentation tasks and have been primarily used for texture classification problems.

Geometric methods assume that texture is composed of a set of basic elements or primatives. Tuceryan and Jain [21] showed how basic texture elements can be extracted from an image using the Vonoroi tessellation of the image. Once these texture elements are extracted, texture features can be computed from their distribution and orientation.

We introduce a method of texture analysis that examines the structure of weighted finite automata that are encodings of the original image. We shall show that this method addresses many of the disadvantages of the aforementioned methods. In Section 2 we review the basic definitions relating to WFA and WFA image compression. Section 3 describes our new texture analysis method and Section 4 presents some experimental results using this method. We conclude with a summary of results in Section 5.

2 Preliminary Defintions and Background

Let $\mathbb{R}$, and $\mathbb{N}$ denote the sets of real and natural numbers respectively. Let $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

Let $\Sigma = \{a_1, a_2, \ldots, a_n\}$ be a finite alphabet. Let Σ^* be the set of all finite strings (words) over Σ. If $n \in \mathbb{N}_0$, Σ^n denotes all words over Σ of length n. The empty word is denoted by λ. The length of a word $w \in \Sigma^*$ is denoted $|w|$.

For a language $L \subseteq \Sigma^*$ and $n \in \mathbb{N}$,

$$\text{pref}_n(L) = \{w \mid w \in \Sigma^n, w\Sigma^* \cap L \neq \emptyset\}$$

is the set of all *prefixes of length n of L* and

$$\text{pref}(L) = \bigcup_{n \geq 0} \text{pref}_n(L)$$

is the set of all *prefixes of L*. We use the short from $\text{pref}(w)$ to denote $\text{pref}(\{w\})$ for $w \in \Sigma^*$.

To avoid confusion between a tuple and an open interval on a number line, we shall use angle brackets for all tuples. Thus the point $x = 0$, $y = 1$ on the Cartesian plane would be denoted $\langle 0, 1\rangle$ and the open interval between 0 and 1 on the real line would be denoted $(0, 1)$. Parameters of functions are enclosed in parentheses as usual.

A greyscale image $\mathcal{I}$ is a set of points over the real space $\mathbb{I} = [0, 1) \times [0, 1) \subseteq \mathbb{R} \times \mathbb{R}$. Each of these points is assigned a grey value from the interval $[0, 1] \subseteq \mathbb{R}$. A grey value of 0 means "black" and a grey value of 1 means "white" with intermediate values being shades of grey. The point $\langle 0, 0\rangle$ is the bottom left corner of the image. The function $\gamma : \mathbb{R} \times \mathbb{R} \to [0, 1] \subseteq \mathbb{R}$ associates each point in $\mathbb{I}$ with its grey value.

In practice the image space $\mathbb{I}$ must be a set of points with rational coordinates because points with irrational coordinates cannot be addressed by a finitely long w. Points with one or more irrational coordinates can only be addressed by infinitely long non-ultimately-periodic words as summarized by Eramian [22].

A weighted quadtree is a quadtree in which every node has an associated real-valued weight. A greyscale image quadtree $\mathcal{Q}$ is a complete weighted quadtree of height n where every node is present up to and including level n. Such a tree is said to specify an image at *resolution* n (an image with pixel dimensions of $2^n \times 2^n$). Each node is assigned an address that is a word over the alphabet $\Sigma = \{0, 1, 2, 3\}$. Addresses are assigned to each node recursively in the following way:

- The root of the tree is assigned the empty word λ.
- If the word w is the address of a node, then the words $w0$, $w1$, $w2$ and $w3$ address the first, second, third and fourth children of that node, respectively.

We write $d(\mathcal{Q}, w)$ to denote the node of $\mathcal{Q}$ that is addressed by w. Every node corresponds to a finite subsquare of $\mathcal{I}$. A subsquare is a square that is closed on the left and bottom sides, and open on the top and right sides. [2] A subsquare is given by a size l (the length of its side) and a center point c, formally a pair $\langle l, c\rangle \in [0,1] \times \mathbb{I}$. Words over Σ^*, as above, address a restricted set of subsquares.

Let $s(w)$ be the subsquare corresponding to $d(\mathcal{Q}, w)$. The notation $s(w)$ can also be read as "the subsquare of $\mathbb{I}$ addressed by w". We recursively define s as follows.

$s(\epsilon) = \langle 1, \langle \frac{1}{2}, \frac{1}{2}\rangle\rangle$.
If $s(w) = \langle l, \langle x, y\rangle\rangle$ then
  $s(w0) = \langle \frac{l}{2}, \langle x - \frac{1}{4}l, y - \frac{1}{4}l\rangle\rangle$ (the bottom left quadrant of $s(w)$),
  $s(w1) = \langle \frac{l}{2}, \langle x - \frac{1}{4}l, y + \frac{1}{4}l\rangle\rangle$ (the top left quadrant of $s(w)$),
  $s(w2) = \langle \frac{l}{2}, \langle x + \frac{1}{4}l, y - \frac{1}{4}l\rangle\rangle$ (the bottom right quadrant of $s(w)$), and
  $s(w3) = \langle \frac{l}{2}, \langle x + \frac{1}{4}l, y + \frac{1}{4}l\rangle\rangle$ (the top right quadrant of $s(w)$).

We slightly abuse our notation and write $\langle x, y\rangle \in s(w) = \langle l, \langle c_x, c_y\rangle\rangle$ if the point $\langle x, y\rangle$ is a member of $[c_x - \frac{l}{2}, c_x + \frac{l}{2}) \times [c_y - \frac{l}{2}, c_y + \frac{l}{2})$, that is, $\langle x, y\rangle$ is within the subsquare $s(w)$. If $P \subseteq \mathbb{I}$ is a set of points then we may also, for some $w \in \Sigma^*$, write $P \subseteq s(w)$ if $\langle x, y\rangle \in s(w)$ for every $\langle x, y\rangle \in P$.

Thus a word over Σ^* is the address of a node in the quadtree $\mathcal{Q}$ as well as the address of a subsquare of size $2^{-|w|}$. We have yet to discuss the weights assigned to each quadtree node. For any $w \in \Sigma^*$ we denote by $g(\mathcal{Q}, w)$ the weight associated with $d(\mathcal{Q}, w)$. We let

$$g(\mathcal{Q}, w) = \frac{1}{l^2} \int_{y-\frac{l}{2}}^{y+\frac{l}{2}} \int_{x-\frac{l}{2}}^{x+\frac{l}{2}} \gamma(x, y)\, dx\, dy$$

where $\langle l, \langle x, y\rangle\rangle = s(w)$. Thus $g(\mathcal{Q}, w)$ is the average grey value of all of the points in $s(w)$.

Definition 1 *(Culik [1]) Let $\Sigma = \{a_1, a_2, \ldots, a_n\}$ be a finite alphabet of n letters. A function $F : \Sigma^* \to \mathbb{R}$ is an average preserving function if for all $w \in \Sigma^*$*

$$F(w) = \frac{1}{n}\left(F(wa_1) + F(wa_2) + \cdots + F(wa_n)\right).$$

[2] Partially open subsquares are necessary to avoid the same point being present in multiple subsquares. See Eramian [22] for further details.

For a quadtree $\mathcal{Q}$, if g is an average preserving function, then $\mathcal{Q}$ defines a *multiresolution greyscale image*. It is clear that g as defined above will always be an average preserving function.

Definition 2 *(Culik [1]) A weighted finite automaton (WFA) is a tuple* $A = \langle Q, \Sigma, \Delta, \alpha, \beta \rangle$ *where*

Q *is a finite set of states.*
Σ *is a finite alphabet*[3]
$\Delta : Q \times \Sigma \times Q \to [-\infty, \infty]$ *is the transition or weight function.*
$\alpha : Q \to [-\infty, \infty]$ *is the initial distribution.*
$\beta : Q \to [-\infty, \infty]$ *is the final distribution.*
If $\Delta((p, a, q)) = g \neq 0$ *then there is a transition from state* p *to state* q *with label* a *and weight* g*.*

The initial and final distributions assign to each state an *initial weight* and a *final weight* respectively. These are analogous to the initial state and final states in a classical finite automaton except that here, each state can be initial or final with a certain weight.

The WFA A defines the function $\varphi_A : \Sigma^* \to [-\infty, \infty]$ such that

$$\varphi_A(w) = \sum_{q_0,\ldots,q_n \in Q} \alpha(q_0) \cdot \Delta(q_0, a_1, q_1) \cdot \Delta(q_1, a_2, q_2) \cdot \ldots \cdot \Delta(q_{n-1}, a_n, q_n) \cdot \beta(q_n)$$

where $w = a_1 a_2 \cdots a_n$. Intuitively, to obtain $\varphi_A(w)$ we compute the sum of the weights of all paths in A whose labels form the word w where the weight of a path is the product of the weights of the transitions on the path, the initial distribution of the first state on the path, and the final distribution of the last state on the path.

In WFA image compression (encoding) algorithms, the automata are constructed so that the function φ_A is identical to, or closely approximates, the function g for the given image quadtree.

To regenerate the original image from an automaton A at resolution n we run all words $w \in \Sigma^n$ on A. A point $\langle x, y \rangle$ in the output image is assigned the grey value $\varphi_A(w)$ if and only if $\langle x, y \rangle \in s(w)$. Thus, each word of length n addresses a pixel of the image at resolution n which contains all of the points in the subsquare $s(w)$.

Additional information on image/automata encoding and decoding algorithms and their variations can be found in papers by Culik [1,3], Katritzke[5], Litow and de Vel [6,23–26], Hafner [7], and Lin [4]. The variation used for our experiments is based on an algorithm due to Culik and can be found in Eramian [22].

[3] For images we use the alphabet $\Sigma = \{0, 1, 2, 3\}$.

In the next section we present a method of texture analysis that operates on the weighted automaton that encodes an image. It is the first texture analysis method to use weighted automata and it demonstrates well the potential for extracting features from the WFA-encoding of an image. We shall see that it also addresses some of the shortcomings of other types of methods.

3 Texture Analysis with WFA

The method we will arrive at computes a texture feature which we will call *texture variance*. We will use a very simple texture model where we say that if a there is a large variation in the greylevels of pixels in a small region, then that region is highly textured; if there is a low variation in greylevels of pixels in a small region, then the region is highly untextured. Using this very simple model we can achieve some very promising results. Our first definition identifies a relationship between greylevel distribution at different resolutions.

Definition 3 *Given a weighted finite automaton* $A = \{Q, \Sigma, \Delta, \alpha, \beta\}$ *for which* $\varphi_A(w) = g(\mathcal{Q}, w)$ *for some greyscale image quadtree* $\mathcal{Q}$ *for greyscale image* $\mathcal{I}$ *with* Σ *the usual quadtree alphabet* $\{0, 1, 2, 3\}$*, the successor variance* $v : \Sigma^* \to \mathbb{R}$ *of a word* $w \in \Sigma^*$ *is*

$$v(w) = \sum_{a_1, a_2 \in \Sigma, a_1 \neq a_2} |\varphi_A(wa_1) - \varphi_A(wa_2)|.$$

Note that $v(w) \in [0, 4] \subseteq \mathbb{R}$. The value of $v(w)$ is maximal when two of the subquadrants of $s(w)$ are totally white ($\varphi(wa_1) = \varphi(wa_2) = 1$), and the other two subquadrants of $s(w)$ are totally black ($\varphi(wa_3) = \varphi(wa_4) = 0$). This gives a sum of $v(w) = (1+1+1+1+0+0) = 4$. The value of $v(w)$ is minimal when each subsquare of $s(w)$ has the same greylevel, which gives $v(w) = (0 + 0 + 0 + 0 + 0 + 0) = 0$. The function v measures how unevenly the greyness of the subsquare $s(w)$ is distributed into its subquadrants $s(w0), s(w1), s(w2), s(w3)$. The more unevenly the greyness is distributed, the more textured the original image within the subsquare $s(w)$.

To obtain local texture information about the subsquare addressed by a word w, we first compute a vector V, which we call the *variance history*, such that $V_i = v(\text{pref}_i(w))$. The vector V holds the variances for every prefix of w so that the large-scale variances are at low indices of V and the small-scale variances are at the larger indices.

We now introduce the concepts of *texture scale* and *texture variance* which are defined in terms of V.

Definition 4 *Given V computed from w, we define the local texture scale about w as:*

$$\sigma(w) = \min\{i \mid V_{i+1} > V_i\}.$$

The intuition here is that, as we begin zooming in on a pixel, the variance should decrease until we are looking at a subsquare where, in the original image, there is a homogeneous texture. It is clear that if a subsquare $s(w)$ of the original image contains a homogeneous texture then, at resolution $|w|$, $v(w)$ will be small. If we now continue to zoom in, the variance should begin increasing again as we begin to focus on the details of the texture region itself. So we define the scale of the texture to be the resolution at which we stop seeing a decrease in variance, and thus should be indicative of the size of a region of homogeneous texture.

We next define texture variance, which will be our measure of how textured are the homogeneous texture regions that we find.

Definition 5 *Given V computed from w, local texture variance is defined as:*

$$\tau(w) = \max\{V_i \mid i > \sigma(w)\}.$$

Once we have identified the texture scale, we find the largest V_i such that i is greater than the texture scale $\sigma(w)$. This should identify scale at which the region is most textured. We color the output image $\mathcal{O}$ according to the value of $\tau(w)$ such that each for each $\langle x, y\rangle \in s(w)$ we let

$$\gamma(x, y) = \log_5(\tau(w) + 1).$$

The resulting map of texture variance should give a good analysis of textural complexity.

Experiments showed quickly that the given definition of σ does not result in a good measure of texture scale since it is clearly possible to have that $\sigma(\epsilon)$ be slightly less than $\sigma(\text{pref}_1(w))$. This could falsely tell us that an entire quadrant of the of the original image is homogeneous in texture when, in fact, it is not. Our experiments have shown that finding the global minimum of V captures the idea of texture scale in a far superior manner. If more than one element of V are global minima, then we choose the one with smallest index. We thus redefine σ as follows:

Definition 6 *Given V computed from w, the local texture scale about w is*

$$\sigma(w) = \min\{i \mid V_i = \min\{V_j \mid 1 \leq j \leq |w|\}\}$$

This definition ensures that we obtain the scale that exhibits the least successor variance, and hence should be representative of the scale of a very homogeneous texture region. This second definition produces results that are far superior to the first definition. In the next section we show experimental results consisting of texture scale and texture variance maps of a few natural images.

4 Experiments

The measures σ and τ can be computed concurrently with the rendering of the original image from the automaton at any given 2^n by 2^n resolution. The successor variance vector V is computed as each input letter of each pixel address is read. Once a complete word w has been processed, $\sigma(w)$ and $\tau(w)$ can be easily computed and stored.

A tunable parameter of the implementation is that we can select which elements of V are considered when computing $\sigma(w)$. In the experiments that follow, we compute $\sigma(w)$ using only elements $V_1, V_2, \ldots, V_{n-1}$ where n is the output resolution. The smallest subsquare's successor variances are left out of the selection because often this allows a smooth area of a microtexture within a homogeneous texture region to be selected as the texture scale. This is not desirable and by not considering V_i in the computation of $\sigma(w)$ we achieve improved results.

Consider the texture scale map in Fig. 3 and the texture variance map in Fig. 4 (the original image is shown in Fig. 2). The false color scale used in the texture variance maps is shown in Figure 1. In the texture scale maps, darker shades of grey represent large scale and lighter shades represent small scale. We see that in the texture variance map (Fig. 4) the streams of water from the fountain, a complicated texture, show up as yellows and oranges indicating a fairly highly textured region as one would expect. The edges in the marble of the fountain wall also show up as highly textured as they should. We note that the face of the fountain wall shows up as a light blue which we expect since it is fairly smooth, but it is not as smooth as the two main fountains which show up as large areas where all of the pixel values are white. We should thus expect that the two large areas of water should be detected as extremely untextured and they are – showing up as a dark blue. We can see from the texture scale map (Fig. 3) that most of the image has very small scale texture, and that, except for the large main water streams, we are primarily discriminating based on microtexture.

Now consider the next image in Fig. 5 and its texture scale and texture variance maps (Fig. 6 and Fig. 7 respectively). We see that we are detecting some

larger areas of homogeneous texture in this image, particularly on the mountain in the background and the rocks and shrub in the foreground. In the texture variance map we see that the sky appropriately registers as the least textured region (dark blue). The edges of the various overlapping slopes are picked up well showing up as sloping yellow-green lines, and the areas of trees and shrubs are denoted as moderately textured, showing up as light blues and greens. The rocks in the foreground register as the most higly textured portion of the image which seems appropriate.

Finally consider our third example shown in figures 8, 9, and 10. Here we see that the relatively large expanses of sky show up as larger scale textures in the texture scale map, the stonework of the towers are primarily a small scale texture and the street in the foreground is primarily an intermediate scale texture. We also see some anomalous regions where the texture scale is large (small regions of black) indicating room for improvement of this method. In this image we have, for the most part, fairly smooth textures with boundaries between them. This is reflected by the fact that the most highly textured areas are the edges between these regions. This is desirable, since it shows that our method has the potential to not only discover two-dimensional features, like in the previous image, but also one-dimensional features like in this image, and our first example image.

5 Conclusions

All of the examples in the previous section were computed in a completely unsupervised setting and the only input to the analysis software was the source image. This gives our method definite advantages over those that require *a-priori* information or supervision, such as model-based and wavelet-based methods, and to a lesser extent region-based methods. Our method does not depend on crisp boundaries between texture regions which is another advantage over region-based and edge-based methods. Our method could be classified as a transform-domain method because we do convert images to the WFA domain, and back. Although the decompression is very fast, which is an advantage over Fourier methods, the compression process can be quite slow.

While disadvantages of some other methods are overcome, the method presented here has its own shortcomings. The quadtree decomposition of the image means that we are only able to examine a small subset of all possible subsquares. There may be features of the image that are undetectable because they either lie precisely on a subsquare boundary or straddle a large subsquare's boundary. Moreover, consider two highly textured regions, one where the greylevel variation is high, but the distribution of greylevels is very regular, and a texture where the variation is equally high, but the distribu-

tion of greylevels is quite irregular. This method cannot tell the difference between these two textures and will attribute an approximately equal amount of texture to each region. Finally, while it is true that if a subsquare has low successor variance it should contain a homogeneous texture, the converse is not necessarily true. Thus it is possible for a highly textured region to appear to have a very low successor variance when averaged out at a particular scale which can result in an inappropriate value of σ. A more sophisticated texture model is required to overcome these difficulties and will be the subject of future research.

Nevertheless, given the straighforwardness of the model, we submit that this method performs remarkably well and is a good general texture analysis algorithm which is suitable for any input image and has certain advantages over known techniques in general. We do not make any claims to its superiority over other specific methods, but do claim that it should perform reasonably well on any image. We also feel that this experiment shows the potential of weighted finite automata based methods in the area of texture analysis especially given that it is extremely easy to perform a large number of affine transformations on images that are encoded as WFA without decompressing (Culik, [27]) – a property which could be taken advantage of to develop some very sophisticated methods. There remain certain open questions to be followed up upon including to what degree is the ability to characterize texture affected by the compression rate, to identify patters in the automata themselves that are characteristic of certain types of texture, and to develop ways of extracting spatial relationships between objects in the image from the automaton representation.

References

[1] K. Culik II, J. Kari, Image compression using weighted finite automata, Comput. and Graphics 17 (3) (1993) 305–313.

[2] E. S. Santos, Maximin automata, Information and Control 13 (1968) 363–377.

[3] K. Culik II, J. Kari, Inference algorithms for WFA and image compression, in: Y. Fisher (Ed.), Fractal Image Encoding and Analysis, Springer-Verlag, 1998.

[4] Y. Lin, H. Yen, An ω-automata approach to the compression of bi-level images, Electronic Notes in Theoretical Computer Science 31 (1).
URL `http://www.elsevier.nl/locate/entcs/volume31.html`

[5] F. Katritzke, Refinements of data compression using weighted finite automata, Ph.D. thesis, University of Siegen (2001).

[6] B. Litow, O. de Vel, On the basic parameters of automaton-based image compression, Tech. Rep. 99/04, James Cook University of North Queensland

(1999).
URL citeseer.nj.nec.com/article/litow99basic.html

[7] U. Hafner, Asymmetric coding in (m)-WFA image compression, Tech. Rep. 132, Lehrstuhl für Informatik II, U. Wuerzburg (1995).
URL citeseer.nj.nec.com/hafner95asymmetric.html

[8] M. Tuceryan, A. K. Jain, Texture analysis, in: C. Chen, L. F. Pau, P. S. P. Wang (Eds.), The Handbook of Pattern Recognition and Computer Vision (2nd edition), World Scientific Publishing Co., 1998, pp. 207–248.

[9] L. V. Gool, P. Dewaele, A. Oosterlinck, Survey: Texture analysis anno 1983, Computer Vision, Graphics and Image Processing 29 (3) (1985) 336–357.

[10] T. Reed, J. M. Hans Du Buf, A review of recent texture segmentation and feature extraction techniques, CVGIP: Image Understanding 57 (3) (1993) 359–372.

[11] R. M. Haralick, Image texture survey, in: P. R. Hrishnaiah, L. N. Kanal (Eds.), Handbook of Statistics, Vol. 2, 1982, pp. 399–415.

[12] T. Reed, M. Werman, Texture segmentation using a diffusion region growing technique, Pattern Recognition 23 (9) (1990) 953–960.

[13] S. Hojjatoleslami, J. Kittler, Region growing: A new approach (1995).
URL citeseer.nj.nec.com/hojjatoleslami95region.html

[14] A. Khotanzad, J. Chen, Unsupervised segmentation of images by edge detection in multidimensional features, IEEE Trans. Pattern Anal. Mach. Intell. 11 (4) (1989) 414–421.

[15] N. Paragios, R. Deriche, Coupled geodesic active regions for image segmentation: A level set approach, in: ECCV (2), 2000, pp. 224–240.
URL citeseer.nj.nec.com/paragios99coupled.html

[16] H. Derin, H. Elliot, Modeling and segmentation of noisy and textured images using Gibbs random fields, IEEE Trans. Pattern Anal. and Mach. Intell. 9 (1).

[17] S. Krishnamachari, R. Chellappa, Multiresolution Gauss-Markov random field models for texture segmentation, IEEE Trans. Image Processing 6 (2).

[18] M. Unser, Texture classification and segmentation using wavelet frames, IEEE Trans. Image Processing 4 (11).

[19] M. G. Eramian, R. A. Schincariol, L. Mansinha, R. G. Stockwell, Generation of aquifer heterogeneity maps using two dimensional spectral texture segmentation techniques, Mathematical Geology 31 (3) (1999) 327–348.

[20] R. M. Haralick, K. Shanmugam, I. Dinstein, Textural features for image classification, IEEE Trans. Syst., Man Cybernet. 3 (1973) 610–621.

[21] M. Tuceryan, A. K. Jain, Texture segmentation using voronoi polygons, IEEE Trans. Pattern Anal. Machine Intelligence 12 (2) (1990) 211–216.

[22] M. G. Eramian, Image texture analysis using weighted finite automata, Ph.D. thesis, The University of Western Ontario, London, Ontario, Canada (2002).

[23] B. Litow, O. de Vel, A recursive GSA acquisition algorithm for image compression, Tech. Rep. 97/2, James Cook University of North Queensland (1997).

[24] B. Litow, O. de Vel:, The weighted finite automaton inference problem (1995).

[25] B. Litow, O. de Vel, On digital images which cannot be generated by small generalised stochastic automata, in: Proc. MFCS Workshop on Randomized Algorithms, 1998, pp. 70–77.
URL `citeseer.nj.nec.com/litow98digital.html`

[26] B. Litow, O. de Vel, Generalised stochastic automaton image compression, Tech. Rep. 96/19, James Cook University of North Queensland (1997).
URL `citeseer.nj.nec.com/litow96generalised.html`

[27] K. Culik II, J. Kari, Finite state transformation of images, Comput. and Graphics 20 (1996) 125–135.

Fig. 1. Scale representing degree of texture from “highly untextured” at the dark blue end to “highly textured” on the dark orange end.

Fig. 2. Original Image

Fig. 3. Texture Scale σ of Fig. 2

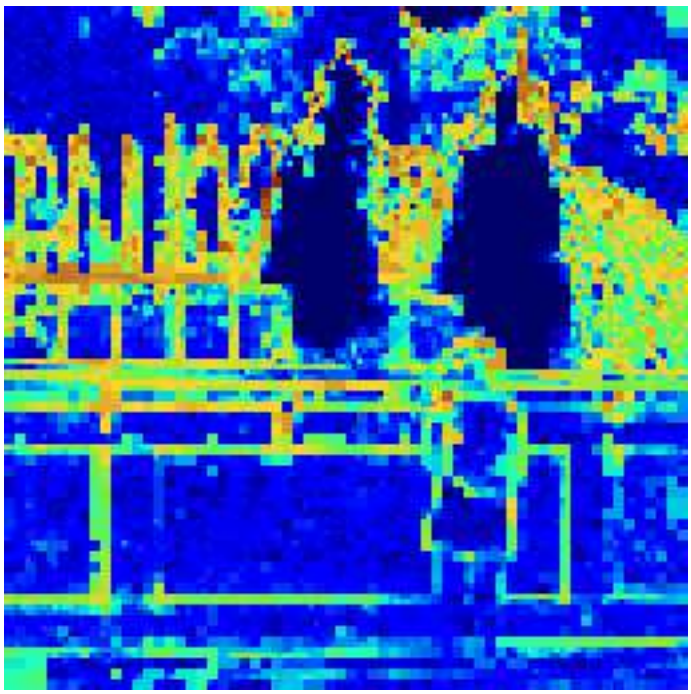

Fig. 4. Texture Variance τ of Fig. 2

Fig. 5. Original Image

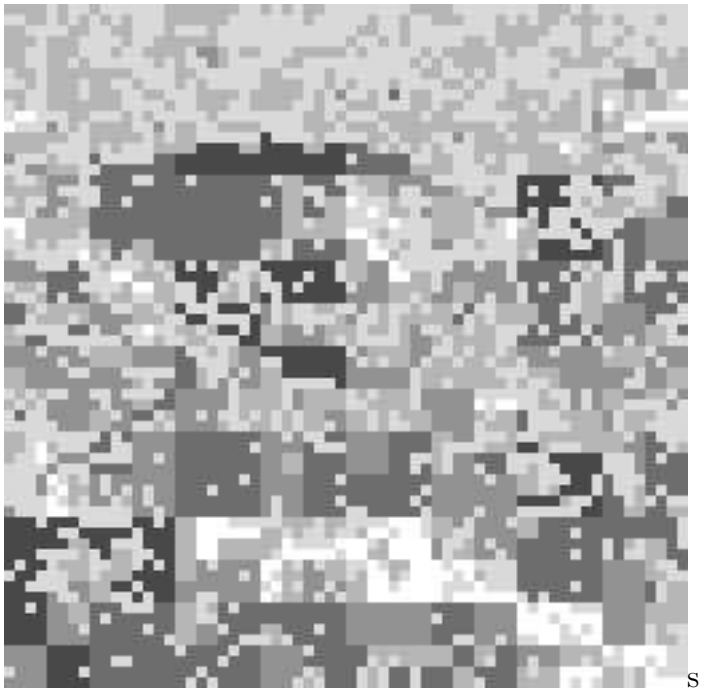

Fig. 6. Texture Scale σ of Fig. 5

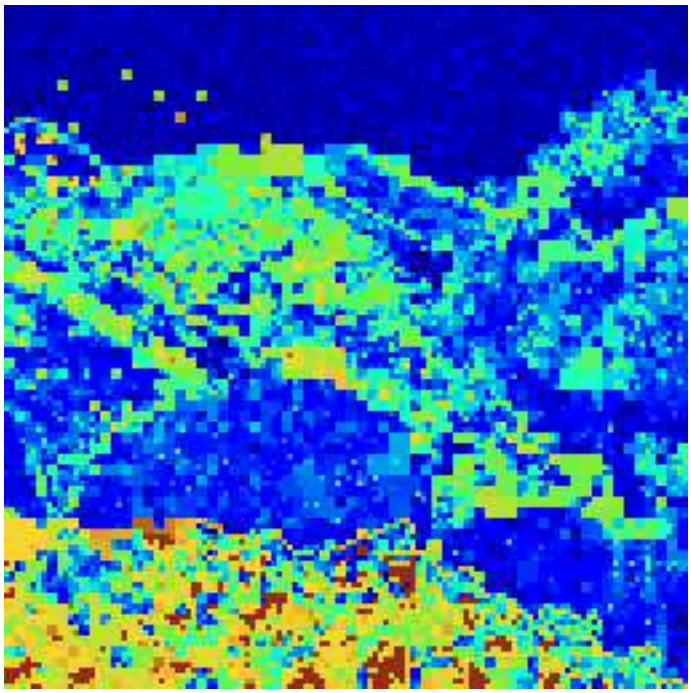

Fig. 7. Texture Variance τ of Fig. 5

Fig. 8. Original Image

Fig. 9. Texture Scale σ of Fig. 8

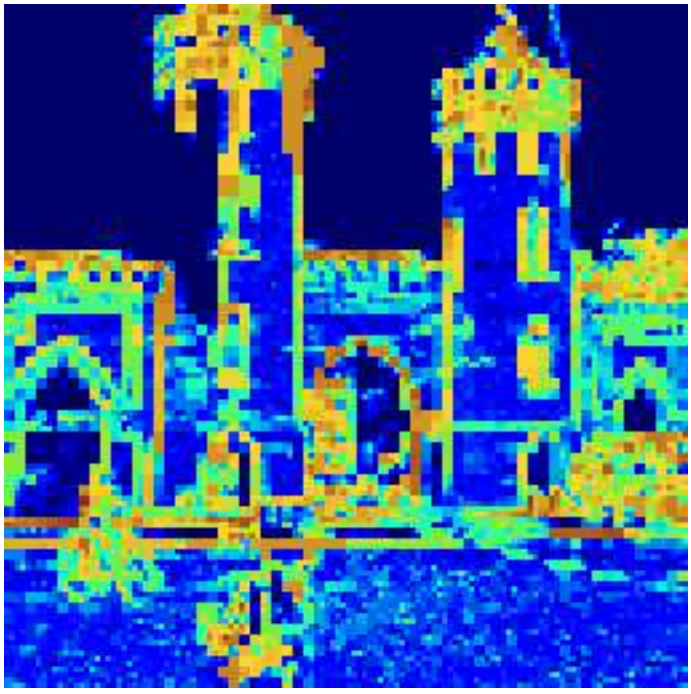

Fig. 10. Texture Variance τ of Fig. 8