UIUCDCS-R-72-531

ANALYSIS OF TEXTURE

By

B. H. McCormick

and

S. N. Jayaramamurthy

July, 1972

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

This work was supported by Contract AT(11-1)-2118 with the U.S. Atomic Energy Commission.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

ACKNOWLEDGMENTS

We wish to express our gratitude to Mr. John S. Read for his participation in the preparation of the paper, "Automatic Generation of Texture Feature Detectors" [3.12], parts of which have been included in this report to preserve continuity. Thanks are also due to Prof. R. S. Michalski for informative and illuminating discussion concerning interval covering theory and use of his PL/1 programs of the A^Q algorithm.

We would like to thank Mrs. Judy Arter, Mrs. Star Starnes, and Mrs. Patti Welch for excellent typing and editing. We also would like to express our appreciation to Mr. Stanley Zundo for the drawings and Mr. Dennis Reed for the offset printing.

TABLE OF CONTENTS

	Page
1. INTRODUCTION.....	1
1.1 Brief Introduction of the Method.....	1
1.1.1 Interval Complexes.....	2
1.1.2 Controlling Parameters.....	2
1.1.3 Distinguishability Index.....	3
1.1.4 Implementation of the Method.....	3
2. LITERATURE SURVEY.....	5
2.1 Visual Texture.....	5
2.2 Models of Texture.....	6
2.2.1 Structural Textures.....	6
2.2.2 Statistical Textures.....	8
3. DESCRIPTION OF THE METHOD.....	10
3.1 Texture Recognition as Statistical Decision Problem.....	10
3.2 Description of the Scheme.....	11
3.2.1 Local Patterns (Events).....	11
3.2.2 Likelihood Ratio.....	12
3.2.3 Decision Goals: Optimal Decision Making.....	14
3.2.4 R.O.C. Curve.....	15
3.3 Classification Scheme.....	17
3.4 Coloring the Scene of Analysis.....	17
3.5 Multiple Textures.....	18
3.6 Texture Discrimination.....	18
4. INTERVAL COVERING THEORY: GENERATION OF TEXTURAL FEATURE DETECTORS.....	20
4.1 Notation.....	20
4.2 Generation of Interval Covers.....	21
4.3 Intervals as Textural Feature Detectors.....	23
5. APPLICATION OF THE METHOD: EXAMPLES.....	30
5.1 Extraction of Textural Regions.....	30
5.1.1 Multiple Regions.....	33
5.2 Border Extraction.....	40
5.3 Iteration.....	48
5.4 Interval Complexes.....	48
6. FUTURE WORK.....	59
6.1 Generalized Linguistic Model.....	59
6.2 Textural Analysis Using Time Series Analysis.....	60
6.2.1 Interpretation of the Results.....	62
6.2.2 "Event" Series: Synthesis of Texture.....	62

	Page
6.3 Optimization of Sampling Strategy.....	63
6.3.1 Quantization Scheme.....	63
6.4 Multiple Textures.....	63
6.5 Analysis of Color Images.....	64
APPENDIX.....	65
Description of the Programs.....	65
TEXROC Program.....	65
NUTEX Program.....	73
BIBLIOGRAPHY.....	77
1. General.....	77
2. Textures--Psychological Aspects.....	78
3. Textures--Methodologies.....	79
4. Textures--Applications.....	81
5. Mathematical Tools.....	82

LIST OF TABLES

Table	Page
1.	28

LIST OF FIGURES

Figure	Page
3.1 Chart Showing Total Number of Events in the Universe For Various Templates.....	13
3.2 Receiver-Operating-Characteristic Curve.....	16
4.1a One Dimensional Texture.....	25
4.1b R.O.C. for Textures Shown in Figure 4.1a.....	26
4.1c Interval Cover for Textures Shown in Figure 4.1a.....	27
4.2 Generalized Logic Diagram with Interval Covering.....	29
5.1a Textured Scenes.....	31
5.1b Template Defining the Event.....	31
5.1c R.O.C. for Textures Shown in Figure 5.1a.....	31
5.2a Samples of Texture T^1 (Various Grid Sizes with and Without Noise).....	34
5.2b Samples of Texture T^0 (Random Pictures).....	35
5.2c R.O.C. for Textures Shown in Figures 5.2a and 5.2b.....	36
5.2d First Set of Test Samples.....	37
5.2e Second Set of Test Samples.....	38
5.2f Results of Classification.....	39
5.3a A Scene with Multiple Testures (t_1, t_2, t_3, t_4).....	41
5.3b Binary "Colored" Image of the Input Scene (Output of the Filter) with $T^1 = \{t_1, t_2\}$ and $T^0 = \{t_3, t_4\}$	42
5.3c Binary "Colored" Image of the Input Scene (Output of the Filter) with $T^1 = \{t_1, t_3\}$ and $T^0 = \{t_2, t_4\}$	43

	Page
5.3d Combination of Figures 5.3b and 5.3c (Resulting in a Scene with Four "Colors").....	44
5.4 Filtered Images of Brain Nuclei.....	45
5.5 Border Extraction: Example 1.....	46
5.6 Border Extraction: Example 2.....	47
5.7a Textures of Straw (T^1) and Wood Grain (T^0).....	49
5.7b "Colored" Images of the Input Pictures at the End of the 5 Iterations.....	50
5.7c Same Textures as Shown in Figure 5.7a Scanned at Coarser Resolution.....	51
5.7d "Colored" Images of Textures in Figure 5.7c at the End of 3rd Iteration.....	51
5.7e Same Output as Figure 5.7d at the End of 5 Iterations.....	52
5.7f R.O.C. Curves for Both (i) Fine and (ii) Coarse Resolution....	52
5.8a Interval Complexes (2-D Filters) for T^1 = Random Texture and T^0 = Herringbone Pattern.....	54
5.8b Multiple Textural Scene (Input Scene).....	55
5.8c Output of One of the Filters with Figure 5.8b as Input.....	56
5.8d Output of Another Filter for Same Input.....	57
5.8e Output of All the Filters Placed in Parallel for the Input Scene Shown in Figure 5.8b.....	58

1. INTRODUCTION

Visual texture is well known to play a central role in visual perception. In the context of automated image processing, there are three problems which have received extensive attention recently, viz., texture discrimination, texture analysis, and texture synthesis. In the literature we find many methods which deal with the above problems. Most of these methods, however, have little or nothing in common and lack generality.

It is our intention to propose and develop a method which would be sufficiently versatile to deal with many of the problems in texture: viz., texture discrimination/recognition, texture analysis, and texture synthesis.

Our proposed method is based on the principles of signal detection theory. By extending this method and with the application of Interval Covering Theory recently developed at the University of Illinois by Professors B. H. McCormick and R. S. Michalski, we show how we generate texture feature detectors.

1.1 Brief Introduction of the Method

To serve as an introduction to the method of texture recognition, we will briefly describe how we presently deal with this problem and later on show how it can be extended to cope with more complex problems.

We often encounter a situation where we are presented with a pair of families of visual scenes (digitized images, of course) which differ in texture, say, those of malignant and nonmalignant tumor cells. The problem is to distinguish and appropriately label any member of these families. This is the problem of texture recognition.

For this purpose, we define a template to extract patterns from the scene of analysis, i.e. n-tuple of gray levels centered around each given positioning of the template. The occurrence of any pattern can be considered as an "event." All possible patterns that can be extracted by the template defines the universe of events. With the help of samples from both texture families, we calculate statistics of each event, such as the probability of its occurrence in any given family, the "likelihood ratio" which is the ratio of the probability of its occurrence in one family to the other, and so on. Using criteria which we shall discuss in detail later in this report, we extract two disjoint sets of patterns (events). The patterns in the same set have the property that their occurrence in one family is more likely than in the other. The classification of an "unseen" sample (i.e. not included in the initial training sample) is achieved on the statistics of the occurrences of patterns belonging to the above-mentioned sets in the scene of analysis. The thresholds needed for the classification are determined with the help of the training samples. We have essentially constructed a filter by defining one set as the passband of events and the other as the stopband. A scene of analysis is classified by the statistics of the events that would fall in these bands.

1.1.1 Interval Complexes

Starting with the aforementioned disjoint sets and with the applications of the methods in "interval covering theory," a set of "interval complexes" are then developed. Reserving the explanation of terminology and other details for a later chapter, we shall add here only that these "interval complexes" serve as 2-D filters for textural features which are more likely to appear in one family than in the other.

1.1.2 Controlling Parameters

It can be seen that the analysis of texture is performed by

using patterns, which are local in nature and can only be as global as the size of the template permits. There are also other parameters other than the size and shape of the template, resolution - the choice of which has a significant effect on the success of the method. Developing an algorithm for an optimal choice of all these parameters, given the scenes of analysis, is one of the top priority items on the agenda for future work.

1.1.3 Distinguishability Index

A visual aid (graph) and an index derived from it has been developed to determine how well the two families of textures can be distinguished for a certain choice of parameters. The graph is known as the R.O.C. (Receiver Operating Characteristic) Curve, as used in Signal Detection Theory. The index can vary from 0.5 for the worst case (two textures are not distinguishable for the present choice of parameters) to 1.0 for an ideal case. This index is a very helpful guide in the iterative choice of a better set of parameters.

1.1.4 Implementation of the Method

Most of the operations in the method suggested can be performed in parallel and it can be very easily implementable on a computer like Illiac III utilizing its parallel processing capabilities. In fact, programs have been written implementing this method in Fortran using "PAX" Subroutines which simulate the Pattern Articulation Unit of Illiac III.

In the sections that follow, we shall discuss some of the problems commonly encountered in dealing with textures. We examine in detail some of the models proposed and recognition strategies suggested in the literature. We then proceed to the formal presentation of our

method and demonstrate some of the earlier successes we have met with.

In the final section we examine the proposal for further work.

2. LITERATURE SURVEY

2.1 Visual Texture

Everyone seems to understand what "texture" means as we live in a world rich in textures. If viewed at an appropriate angle, texture can be seen in almost every scene. Yet it is one of those terms which has escaped a precise scientific definition. As a matter of fact, the host of visual scenes indicated by the term texture is so enormously large and varied in nature, it appears that it is a very difficult task to span the varied concepts of textures by a single definition. The definitions like "texture is that property of material which indicates what it feels like if touched" and adjectives like "rough," "smooth," etc. deal with tactile textures. We are here exclusively interested in visual textures.

Pickett [2.13] observes that for any visual scene to be seen as texture, there should be "a large number of elements (spatial variations in intensity and/or wavelength)" and that "the elements and rules of spacing or arrangement may be arbitrarily manipulated provided a characteristic repetitiveness remains." He adds, "provided there is sufficient detail shown in a small enough visual angle, a characteristic texture emerges even when the basic elements or spacings are randomly distributed." Thus the primary attributes of visual texture are "many variations" and "repetitive variations."

Almost similar aspects are found in other available definitions. But even after a suitable choice of a working definition, there remain many difficulties inherent in the task of texture extraction technology. Hawkins [3.7] in his paper enumerates these in great detail and comes to the conclusion that "texture classification may very well be one of the more difficult tasks in the field of image processing."

Visual texture is really very sensitive to external conditions like lighting and angle of view, etc. The same scene may present a very different texture even for slight modification in the external conditions. Pictures in Brodatz's [3.1] book on "Textures" can be used to endorse the above statement. To simplify the already complicated problem, we assume that the scenes with which we are concerned are taken under similar conditions.

With these problems and its prospective applications in important fields like biomedical sciences and remote sensing of the environment, texture analysis stands as an attractive as well as a challenging problem.

2.2 Models of Texture

Textures can be subdivided into at least two categories: statistical textures and structural textures. Statistical textures in a visual scene can be regarded as defined by a set of statistics extracted from a large ensemble of local measurements made on the scene. We need more information than this to define a structural texture; here the texture is considered to be defined by subpatterns which occur repeatedly within the overall pattern according to well-defined placement rules, as for example, wallpaper.

Though we find many textures which can be classified adequately under one of these asymptotic categories, there are still many textures for which a strict classification may be questionable.

2.2.1 Structural Textures

As its name suggests, the subpattern/placement model [3.15] appears to be the most appropriate model to deal with structural textures. This model is by far very widely used model in the available literature. The "Subpattern" is sometimes referred to as the "unit cell." Usually,

but not necessarily, the subpattern itself might contain sub-sub patterns and so on. For simplicity, only one level of hierarchy is considered here.

As we accept this model, the description of texture is indeed simplified to developing a language to describe the unit cell(s) and spelling out the rules of its placement over a given region. Similarly, synthetic generation can be employed to develop the unit cell(s) and distribute it over the given area according to (appropriate curvilinear) placement rules. The works of Trout [3.22], Conroy [3.4], and Rosenfeld [3.15] reflect a similar treatment of these problems. Trout attempts a formal description of wallpaper-type patterns. The constraints of the wallpaper design are liberalized to provide a formal description of texture on a "generalized wallpaper" design. He develops a syntax-like vehicle to describe primitive patterns. In terms of these patterns and appropriate syntactic rules, the unit cell is described. The wallpaper pattern, and eventually more complex textural patterns, can be described effectively in terms of these unit cells and their distribution. The success of this scheme obviously depends on the judicious choice of the unit cell, primitives, and appropriate syntactic rules. The crux of the problem lies in the choice of the above parameters based on the scene of analysis. This deserves more attention. Without practical simulation one cannot decide how appropriate the descriptive scheme is.

Conroy [3.4] introduces the notion of "signs," defining these to be any collection of lines, usually in close spatial proximity to each other. These are similar to Trout's "primitives." Suitably manipulating the composition of various signs of different sizes, Conroy attempts to generate realistic textures. His method can be used to generate a unit cell and construct the whole scene by properly distri-

buting the unit cell. His program requires as an input the choice of "signs" and their sizes, densities, and appropriate spatial distributions.

Rosenfeld and Lipkin also start with the unit cell/placement rule model for the synthesis of texture. Various problems encountered in this process have been neatly explained. Results are encouraging only when dealing with man-made or artificial textures where the extraction of unit cell is not a difficult problem. When dealing with the real world, for the description or generation of natural texture, the extensions of the above methods have not yielded satisfactory results.

The reason for the failure of these methods when dealing with natural textures is obvious. The extraction of the holy unit cell(s), whose size and shape is unknown, is a formidable task. (So is the choice of primitives, "signs," placement rules, etc. on which the success of the schemes described are very much dependent.) This is the present impediment when dealing with the analysis of texture. Further, when analyzing a piece of texture, even if one has successfully extracted a unit cell, it is yet very difficult to decode the "secret" of the placement rules--which demands a thorough analysis at the global level.

2.2.2 Statistical Textures

Many investigators in this field have developed methods which are usually statistical in nature when working with natural textures. Rosenfeld et al have attempted to discriminate textures by merely detecting the differences in averages of local properties. They demonstrate their scheme by detecting textural edges using the gray level as the local property.

Muerle [3.10] also uses statistical analysis for the discrimination of textures. He divides the whole scene of analysis into many cells. Starting with a single cell, he expands the region by comparing the

statistical distributions of the cell with its neighbors and adding the new cells to fragments which have similar distributions.

There is yet another approach for dealing with problems of texture, suggested by Bartels et al [3.2]. In this scheme a digitized textural scene is regarded as two-way time series. Methods have been developed [5.2] to determine the "order of dependence" in a two-way time series. That is to say, the number of neighboring cells which influence the brightness of a given point can be estimated.

This information and other relevant stochastic properties extracted from the time series have been successfully used for the proper diagnosis of tumor cells by Bartels [3.2].

3. DESCRIPTION OF THE METHOD

When presented with a textural scene for analysis, we need to perform the following tasks. First we have to discriminate various textural regions. This is the task of scene segmentation using textural information. Later, if needed, we may have to identify each textural region with any one of the classes of textures presented before. This is the task of texture recognition. The next in line is the more difficult task: to analyze each homogenous texture and try to extract significant textural features from it. Ideally, of course, we would like to extract minimal information needed, perhaps to faithfully reconstruct or synthesize the original texture. This procedure, when fully developed, would constitute texture synthesis.

3.1 Texture Recognition as Statistical Decision Problem

Texture Recognition can be basically viewed as a statistical decision problem, wherein, like a single trial of a psychophysical experiment, a cycle begins with the presentation of textural scenes and ends with the response of the decision maker. The type of responses expected of the decision maker varies with the problem. For example, in a typical recognition problem, given a textural scene T_x , a decision is to be made if it belongs to T^1 or T^0 . Here T^1 represents a family of visual scenes consisting of a particular texture, or, one of the hypotheses. The alternate is T^0 which does not contain that particular texture. The anticipated response here would be either "1" or "0." This is the binary case we just considered and the decision problem can be extended to a case with multiple hypotheses.

It must be clear that we treated the particular texture as

the "signal." Other textures or even the variations in the same textures can be treated as the noise. When viewed from this angle, texture recognition boils down to the problem of detection of signal in the presence of noise and readily becomes amenable to well-developed methods in signal detection theory.

3.2 Description of the Scheme

Ideally, each texture is composed of one single pattern, "the unit cell" and if one is able to extract it, the texture recognition problem would be reduced to comparing unit cells. But as we have already discussed, things need not be so simple: More than one unit cell may be present in a given scene, the placement rules may be difficult to extract, or no unit cell may be detected.

We avoid these problems by defining our own Universe of local patterns. We try to determine which of these local patterns characterize any given texture. At this point, we use the elements of statistical decision theory. In what follows, we define some of the basic terms needed for further presentation of the method.

3.2.1 Local Patterns (Events)

We define a template centered around a point which samples the patterns from the scene of analysis. Each possible pattern is an n -tuple of gray levels of the nearest n neighbors of the given point and is represented as the n -dimensional vector, e.g. $e = (x_1, x_2, \dots, x_n)$ and is regarded as an "event." All possible local patterns define the Universe of Events, E^U . For example, if the digital picture was quantized to " h " gray levels and the sampling template has n -cells, then there

are h^n elements in E^U . Figure 3.1 shows the number of elements in the universe for the templates shown.

3.2.2 Likelihood Ratio

The "events" described just now serve as the local evidence for the decision maker at the local level. The "likelihood ratio" of an event is a single number which is an indicator of the strength of evidence that the occurrence of that particular event would provide for the presence of the signal. More precisely it is the ratio of the probability of the occurrence of the event in T^1 to that in T^0 . This is estimated from the "training samples" provided from both the families of T^1 and T^0 .

Let E^1 and E^0 be the sets of events obtained from scanning T^1 and T^0 with a given template, respectively. Let

$n_1(e_k)$ = The number of occurrences of event e_k in T^1

$n_0(e_k)$ = The number of occurrences of event e_k in T^0

nT^1 = The number of events in T^1

nT^0 = The number of events in T^0 .

$P(e_k|T^1)$ = The probability of occurrence of e_k in T^1
i.e. probability of e_k conditional on T^1 .

$$= \frac{n_1(e_k)}{nT^1}$$

$P(e_k|T^0)$ is defined similarly.

Then, the "likelihood ratio" of the event e_k is:

$$LR(e_k) = \gamma \frac{P(e_k|T^1)}{P(e_k|T^0)}$$

where " γ " is a normalization factor which compensates for the intrinsic

EVENT		NUMBER OF EVENTS IN THE UNIVERSE (E^U)			
REGULAR	TEMPLATE	NGL = # of GRAY LEVELS IN THE PICTURE			
	HEXAGONAL	NGL = 2	3	4	16
	2	3	4	16
	4	9	16	256
8	27	64	4096
16	81	256	65,536
32	243	1024	1,068,576
64	729	4096	17M
	128	2187	16,384	268M
512	19,683	262,144	68,600M

Figure 3.1. Chart Showing Total Number of Events in the Universe for Various Templates

probabilities.

The "a posteriori probability," that is the probability of truth of the hypothesis at a local level conditional on the occurrence of the event, can be shown to be:

$$P(T^1|e_k) = \lambda(e_k) = \frac{LR(e_k)}{1+LR(e_k)}$$

3.2.3 Decision Goals: Optimal Decision Making

There are four outcomes in the binary signal detection problems, two errors and two correct decisions. A "false alarm" is to choose T^1 when it is T^0 and a "miss" is vice-versa. A "hit" is a correct choice of T^1 , and the correct choice of T^0 is known as "correct rejection." There may be different values associated with the correct decisions and different costs associated with the errors.

There can be many decision goals. One of the goals, for instance, is to maximize the expected value. The "expected value" is the sum of the four terms, each representing the value or cost associated with an outcome weighted by its probability. Other decision goals could be to: maximize a weighted combination, maximize the percentage of correct responses, satisfy the Neyman-Pearson objective, and so on. These are discussed in detail by Green and Swets [5.4] and they prove that whatever is the decision goal, the likelihood ratio criterion is the optimal decision rule, that is to choose T^1 if $LR(e_k) > \beta$, where β is a positive number. β may vary for each decision goal.

With this background, we are ready to partially define local categorizer ψ_R on the basis of training set of information:

Let E^U = event space

$$F^{1\beta} = \{e | e \in E^1 \cup E^0 \text{ and } LR(e) > \beta\}$$

$$F^{0\beta} = \{e | e \in E^1 \cup E^0 \text{ and } LR(e) \leq \beta\}$$

$$F^* = \{e | e \in E^U \setminus E^1 \cup E^0\}$$

Then, define ψ_R by its acceptance set R , i.e.

$$\psi_R(e) = 1 \text{ iff } e \in R$$

$$\text{where } F^{1\beta} \subseteq R \subseteq F^{1\beta} \cup F^*$$

$$\text{and } R \cap F^{0\beta} = \emptyset$$

Note that the determination of which events in F^* are in R has not been made at this point. These represent the "don't-care" events and their assignment is made later in this report.

3.2.4 R.O.C. Curve

The Receiver-operating-characteristic (ROC) curve (Figure 3.2) is a useful device for observing and predicting the behavior of these categorizers. To make the curve, each event $e \in E^1 \cup E^0$ is regarded as a two-component vector with $x = p(e|T^0)$ and $y = p(e|T^1)$. An ordering can be imposed on these vectors by sorting them in descending order by the likelihood ratios of the e 's. The curve is generated by placing the tail of the first vector at the origin and then concatenating the rest in order. The ROC displays several useful items of information in an easy-to-see form. For one thing, the training-set performance of a categorizer for each value of β is shown directly, since for each threshold, the y-coordinate is equal to $\sum_{\{e | LR(e) > \beta\}} p(e|T^1)$ and the x-coordinate is equal to $\sum_{\{e | LR(e) > \beta\}} p(e|T^0)$. The point on the ROC corresponding to a given value of β is easy to find, since it is the

Figure 3.2.. Receiver-Operating-Characteristic Curve

head of the vector with slope = β . (Note that β only has a discrete number of values with different performance effects.) The ROC also provides a measure of the inherent separability of the textures in the training set. The area under the curve is equal to .5 if the textures are non-distinguishable (all events occur with equal probability in both textures) and is equal to 1.0 if the textures are perfectly distinguishable (all events occur in one or the other texture but not both).

3.3 Classification Scheme

As indicated earlier, there are two levels in our classification scheme. One of the simple procedures would be to "mark" all the events belonging to the acceptance set R which occur in the scene of analysis. Based on the global characteristics such as the percentages of these "marks" final decision regarding the state of the world can be made. The thresholds needed for this global classification would be estimated from the learning samples.

We did not include events from F^* in the acceptance set. If the events in R are not too many, this simple method would provide quick and satisfactory results. It becomes cumbersome if there are too many events in R. This case is dealt with in the next chapter.

3.4 Coloring the Scene of Analysis

It is sometimes convenient to have the following transformation of the scene. A point in the transformed plane is marked "1" (or dark) if the corresponding point in the scene of analysis has an event in the acceptance set R. Otherwise, it is marked "0" (light). The resultant picture is the "colored" image of the scene of analysis. An ideal case:

Colored images of samples of T^1 would be all "dark" and those from T^0 would be all "light." This would be the same as the output of a series of filters with the scene of analysis at the input. Generation of these filters (Interval Complexes) from the acceptance set will be described in the next chapter.

3.5 Multiple Textures

Thus far we have considered only binary signal detection (T^1 and T^0). The scheme can be extended to multiple textures case just the same way as the binary signal detection problem is extended to that of M-ary signal detection. When there are M different classes of textures present, $(T_1, T_2, \dots, T_M)$, we need to calculate at least M-1 "likelihood ratios" from which other likelihood ratios can be calculated.

Here
$$LR_{ij}(e_k) = \gamma_{ij} \frac{P(e_k | T_i)}{P(e_k | T_j)}$$

The universe of events will be partitioned into M-disjoint acceptance sets. An event e_k will be included in the R_i acceptance set, iff,

$$LR_{ji}(e_k) > 1 \text{ for all } i \neq j$$

This simple extension sometimes leads into many practical difficulties when dealing with large M for certain choice of parameters. This phase is still under investigation and possible ways to resolve some of the difficulties will be mentioned in the last chapter.

In general for satisfactory results, for larger M, we need to have a larger universe of events, which means more computation time.

3.6 Texture Discrimination

We can mark the various textural regions in a composite scene

if we are given the samples of textures that might be present in it. The procedure is the same as before: extract disjoint acceptance sets of events representing each texture, "color" the composite scene with each set in a different "color" (gray level). Ideally, we end up with each textural region marked with a different appropriate color.

For extracting the "textural features" we need to introduce the elements of interval covering theory which is attempted in the next chapter.

4. INTERVAL COVERING THEORY: GENERATION OF TEXTURAL FEATURE DETECTORS

We have described a local categorizer in the previous section and in principle it could be implemented by just looking up input events in a table of events and likelihood ratios. However, for real textures and useful neighborhood sizes, the number of events in the acceptance set R could be too large to make the process practical. Also, no categorization would be performed for events not in the training set. By applying some concepts from switching theory, equivalent but much more efficient categorizers can be generated. This is accomplished by a technique analogous to switching-theoretic procedures for minimization of the disjunctive normal form of a switching function.

In the case of binary signal detection, the events from $F^{1\beta}$ can be considered a "true" set and those from $F^{0\beta}$ are considered a "false" set. The disjunctive normal form can be expressed as $\bigvee_i \xi_i(e_i)$ where ξ_i is a predicate that has output "true" when the input is a particular event, e_i , from $F^{1\beta}$, and output "false" if it is from $F^{0\beta}$. The symbol $\bigvee$ above represents the logical "OR" of the predicates. Events from F^* are considered as don't-care events. McCormick and Michalski have developed "interval covering theory" as a generalization of switching theory [5.8] which permits the transplantation of much of the minimization machinery already in existence. In particular, Michalski's A^Q algorithm [5.7] for generation of quasi-minimal covers can be used.

4.1 Notation

To explain the method, it is necessary to introduce a few items of notation from [5.7]:

E is the event space as before. That is, the set of all events

$e = (x_1, x_2, \dots, x_n)$ where $0 \leq x_i \leq h - 1$.

A literal, $a_i x_i b_i$, is the set of all events $e \in E$ whose i -th component lies between a_i and b_i :

$$a_i x_i b_i = \{(x_1, x_2, \dots, x_n) \mid a_i \leq x_i \leq b_i\}.$$

An interval is a set-theoretic product of literals,

$$L = \bigcap_{i \in I} a_i x_i b_i, \quad I \subseteq \{1, 2, \dots, n\}.$$

The interval represents a "box" in hyperspace which includes all events between $(a_1, a_2, \dots, a_n)$ and $(b_1, b_2, \dots, b_n)$. Note that components not specified by the interval are free to take on any integer value in $[0, h-1]$.

An interval cover of the set $F^{1\beta}$ against $F^{0\beta}$ is defined as a union of intervals, L_j , such that:

$$F^{1\beta} \subseteq \bigcup L_j \subseteq F^{1\beta} \cup F^*.$$

Thus an interval cover contains all the events in $F^{1\beta}$ plus some in F^* , but none in $F^{0\beta}$. However, the interval cover will represent this partitioning of the space of possible events much more concisely than just enumerating all the events in the acceptance set for T^1 . Also, the interval cover can classify events which were not in the training set, because of the inclusion of F^* events in the "boxes." A quasi-minimal cover can be generated via the A^Q algorithm, which we can only sketch here.

4.2 Generation of Interval Covers

We can make this procedure clear by means of a simple example. For 1-D textures shown in Figure 4.1a, the $F^{1\beta}$ and $F^{0\beta}$ are shown in Table 1. For this example, an interval cover (a minimal one, as it turns out) can be generated manually by means of a visual aid, the Generalized Logic Diagram (GLD), which was introduced by Michalski [5.7] (Figure 4.2).

In the particular case when variables assume only two values, the GLD reduces to a diagram which resembles a Marquand-Veitch diagram. The GLD is a representation of the entire event space, E : 64 events in this case. To use it, the events of $F^{1\beta}$ are mapped in as ones and those of $F^{0\beta}$ as zeros. The squares left over represent F^* (don't cares). The cover is found by an iterative procedure which begins by picking the first "one" encountered in a TV-like scan of the GLD, and discovering all of the maximal intervals which include that "one," but no zeros (an interval "star"). One of these, the interval including the most "ones," is added to the covering set (initially empty). All of the ones included in the "star" are temporarily eliminated, and the scan of the GLD is resumed. The first "one" encountered is selected, and the iteration repeats. Eventually all the "ones" have been eliminated. If all "ones" are included in the covering set, then the cover is minimal. Otherwise, the cover is patched up to include the neglected events, and may not be minimal.

This procedure was followed for the example, and a minimal covering using three intervals resulted: (Figure 4.1b)

$$L_1 = \begin{matrix} 2 & 3 \\ x_2 & x_3 \end{matrix}, \quad L_2 = \begin{matrix} 1 & 3 & 0 & 2 \\ x_1 & x_2 & x_3 \end{matrix}, \quad L_3 = \begin{matrix} 2 & 3 \\ x_1 & x_2 \end{matrix}$$

If these are used to form a categorizer Ψ_R , where $\Psi_R(e) = 1$ for $e \in R = L_1 \cup L_2 \cup L_3$, then the event categorization shown in Figure 4.1 results. The asterisks appearing above Texture 1 and Texture 0 indicate events for which Ψ_R had output = 1. The subscripts on the asterisks denote the interval producing the "hit." Notice that the probability of a hit ($19/20 = .95$) and the probability of a false alarm ($3/20 = .15$) is as predicted by the ROC (Figure 4.1c) for the likelihood ratio decision rule with $\beta = 1$. These textures could easily be discriminated by labeling regions with hit density over some averaging aperture greater

than, say, 55% as Texture 1. In a digital parallel processor, like the Illiac III's Pattern Articulation Unit, Ψ is an image filter. The input to the filter is a digitized picture in several gray values, and the output is a binary plane labeling each element in the input picture as to which texture the picture element most likely belongs. The application of simple smoothing or noise-removal algorithms would then make segmentation into texture regions relatively easy.

4.3 Intervals as Textural Feature Detectors

Intervals were achieved as "boxes" into which events from the acceptance set R were efficiently packed. As the events in the acceptance set occur more frequently in T^1 than in T^0 , the intervals which are nothing but groups of such events, have a tendency to define features which are more likely to be found in T^1 than in T^0 . Sometimes it has been found in a practical case that some intervals pick up features like vertical lines, horizontal lines, herringbone patterns, etc. which are perceived by human beings. Some of the features extracted by other choices of intervals may be very poorly matched for human perception. Strategies for the choice of "good" intervals remains to date largely unexplored.

These "intervals" can be treated as 2-D filters which detect textural features more common to T^1 than T^0 . The normalized "output count" of the filters (normalized number of input "local patterns" from the scene of analysis that fall in the "pass band" of the filter) can be used as a feature vector. In this multidimensional space, we will be dealing with clusters of the scenes from T^1 and T^0 and for classification purposes we can resort to any popular cluster analysis methods.

It may be noted that the generation of "interval covers" was possible for the binary case only. Extension of this procedure for the case of multiple textures is reserved for further investigation.

TEXTURE 1

TEXTURE 0

Figure 4.1a. One Dimensional Texture

Figure 4.1b. R.O.C. for Textures Shown in Figure 4.1a

TEMPLATE DEFINING EVENTS:

INTERVAL COVER

 L_1

 ${}^2X_2^3 \quad X_3^3$
 L_2

 ${}^1X_1^3 \quad {}^0X_2^2 \quad {}^2X_3^3$
 L_3

 ${}^2X_1^3 \quad {}^2X_2^3$

NOTE: Lower limit for each variable is indicated in lower right hand corner and the upper limit in the upper left hand corner.

Figure 4.1c. Interval Cover for Textures Shown in Figure 4.1a

Table 1

Event (e)		n_1	$p(e T_1)$	n_0	$p(e T_0)$	LR(e)
(1,2,3)	3	.15	0	0	0
	(2,3,2)	5	.25	0	0	0
	(2,1,2)	2	.10	0	0	0
	(1,0,2)	1	.05	0	0	0
	(0,2,3)	2	.10	0	0	0
	(3,2,0)	1	.05	0	0	0
	(2,0,2)	1	.05	0	0	0
(3,2,1)	4	.20	3	.15	1.33
	(2,1,0)	1	.05	4	.20	.25
	(1,3,2)	0	0	3	.15	0
	(1,0,1)	0	0	4	.20	0
	(0,1,2)	0	0	2	.10	0
	(1,2,1)	0	0	2	.10	0
	(0,1,3)	0	0	2	.10	0

Figure 4.2. Generalized Logic Diagram with Interval Covering

5. APPLICATION OF THE METHOD: EXAMPLES

For the purpose of identification, each event can be assigned a unique positive number. It will be convenient to refer to the event by this number known as the "Gamma" of the event.

$$e = \{x_1, x_2, \dots, x_n\}$$

$$\gamma(e) = \sum_{i=0}^{n-1} x_{n-i} h^i$$

where h is the no. of gray levels into which the digital picture is quantized.

We introduce the following three disjoint sets which will be very frequently referred to in the examples:

$$F^1 = \{e | e \in E^1 \text{ and } e \notin E^0\} = \{e | e \in E^1 \cup E^0 \text{ and } LR(e) = \infty\}$$

$$F^0 = \{e | e \notin E^1 \text{ and } e \in E^0\} = \{e | e \in E^1 \cup E^0 \text{ and } LR(e) = 0\}$$

$$F^\phi = \{e | e \in (E^1 \cup E^0) \setminus (F^1 \cup F^0)\}$$

The events in F^ϕ are arranged in descending order according to the likelihood ratio.

5.1 Extraction of Textural Regions

We will first illustrate the method with a very simple example.

T^1 is the checkerboard pattern, and T^0 is the unit grid pattern.

(Fig. 5.1a). Both are binary textures.

The template which extracts events is shown in Fig. 5.1b. Total number of events in the universe is $2^4 = 16$.

We scan the input pictures. Collect E^1 and E^0 . We can partition E^u into the following disjoint sets.

$$F^1 = \{10, 5\}^*$$

$$F^\phi = \{ \}$$

$$F^0 = \{14, 13, 11, 7\}^*$$

CHECKERBOARD PATTERN

 T^1

GRID PATTERN

 T^0

Figure 5.1a. Textured Scenes

g_2	g_1
g_3	g_0^*

$$e = \{g_0, g_1, g_2, g_3\}$$

Figure 5.1b. Template Defining the Event

Figure 5.1c. R.O.C. for Textures Shown in Figure 5.1a

$F^* =$ Rest of the 10 events

As there are no common events, we have $F^{1\beta} = \{10, 5\}$ and $F^{0\beta} = \{14, 13, 11, 7\}$ for all β . The ideal R.O.C. curve is shown in Figure 5.1c.

It can be clearly seen that we are able to extract the "unit cells" from the above scenes with the size of the template defined; this is the secret behind the ideal separation. Now it is easy to identify these regions in a combined scene by recognizing the events. A corresponding point in an output plane is marked "1" or "0" depending on whether the event extracted at a given point in the combined scene belongs to either $F^{1\beta}$ or $F^{0\beta}$ respectively. Thus we obtain a "colored" output picture with each region marked differently. We can even use this plane as a mask to extract the desired region from the original scene.

Generally, the separation will not be ideal, in which case we do not expect that any region will appear in a single color. In such cases we use training samples for both T^1 and T^0 and color them. Then we extract a threshold for the area of a given region to be in one color (or for the ratio of areas of different colors) which classifies the majority of the training samples correctly. We can color the unknown samples and will classify them by using this threshold.

Example:

Samples of textures T^1 and T^0 are shown in the figs. 5.2a and 5.2b. These are binary pictures, and a 3 x 2 sampling template is used. The R.O.C. is shown in Fig. 5.2c.

We used grids of the same sizes as used in T^1 but with different amounts of noise added in the first set of test samples (Figure 5.2d) to be recognized as against some random pictures. A correct classification has been made for this set as

* Here onwards events are identified by γ 's only. N-tuple representation will only be shown when clarity is needed.

seen in Figure 5.2f. In the second set (Figure 5.2e) we used grids of larger sizes with and without noise. Still the classification was correct except in the presence of excessive noise.

5.1.1 Multiple Regions

The above procedure can be extended to deal with multiple regions. The output picture is expected to have at least as many gray-levels (colors) as the number of regions present in a combined scene. Let $t_1, t_2, \dots, t_m$ be the protosamples of m different textures. We can assign an n -digit binary number to each sample, where $2^n \geq m$. We shall start with the least significant bit (LSB). All the samples with "1" in the last bit are grouped under T^1 and those with "0" under T^0 . For this pair of textures we obtain an output plane in binary colors as before. This one corresponds to the least significant plane in the output stack.

We can repeat this procedure for all the bits at which time the output stack contains, for an ideal case, each region marked in a different gray level (which is the same as the binary number assigned to that region).

The success of this scheme for any given m depends on the nature of the template (size and shape) selected. (Here we assume that the number of gray-levels in a scene and the resolution are invariant. The influence of these parameters on the said scheme is a case for further investigation). The template size should be large enough to accomodate the largest unit cell in the given samples. Normally better results are expected with larger templates, the only drawback being the loss of sharpness of the boundary curves.

The basic limitation of this scheme is that the operations on the pictures can only be as global as the size of the template permits.. That is to say, for smaller templates the analysis is pretty much local in nature. If, in any case, analysis at a more global level becomes necessary, this

Figure 5.2a. Samples of Texture T₁
(Various Grid Sizes with and without Noise)

Figure 5.2b. Samples of Texture T^0
(Random Pictures)

Figure 5.2c. R.O.C. for Textures Shown in Figure 5.2a and Figure 5.2b

[illegible][illegible][illegible][illegible][illegible][illegible][illegible]

Figure 5.2d. First Set of Test Samples

Figure 5.2e. Second Set of Test Samples

***** CLASSIFICATION *****		
SAMPLE #	::= 1	TEXTURE ::= 0
SAMPLE #	::= 2	TEXTURE ::= 1
SAMPLE #	::= 3	TEXTURE ::= 0
SAMPLE #	::= 4	TEXTURE ::= 1
SAMPLE #	::= 5	TEXTURE ::= 1
SAMPLE #	::= 6	TEXTURE ::= 1
SAMPLE #	::= 7	TEXTURE ::= 0
SAMPLE #	::= 8	TEXTURE ::= 1
SAMPLE #	::= 9	TEXTURE ::= 0
***** CLASSIFICATION *****		
SAMPLE #	::= 1	TEXTURE ::= 1
SAMPLE #	::= 2	TEXTURE ::= 1
SAMPLE #	::= 3	TEXTURE ::= 0
SAMPLE #	::= 4	TEXTURE ::= 1
SAMPLE #	::= 5	TEXTURE ::= 1
SAMPLE #	::= 6	TEXTURE ::= 0
SAMPLE #	::= 7	TEXTURE ::= 1
SAMPLE #	::= 8	TEXTURE ::= 1
SAMPLE #	::= 9	TEXTURE ::= 0
ENTER ROC		

} Test Samples Set 1

} Test Samples Set 2

Figure 5.2f. Results of Classification

scheme may be conveniently used as a preprocessor.

Example

Fig. 5.3a is the input picture, consisting of four different textures t_1 , t_2 , t_3 , and t_4 . (Grids of various sizes.) Fig. 5.3b is the binary "coloured" picture $\{t_1, t_2\}$ as T^1 and $\{t_3, t_4\}$ as T^0 . Fig. 5.3c is the binary "coloured" with $\{t_1, t_3\}$ as T^1 and $\{t_2, t_4\}$ as T^0 . The combined picture (Fig. 5.3d) shows each region marked in different gray level.

Example

Fig. 5.4 shows the brain cells with three textures, background, cytoplasm and nucleus marked differently.

5.2 Border Extraction

When two textured regions meet, the events picked up along the border can be expected to be different from those that occur in the core of either regions, for the simple reason that they are made up from parts of both the regions. We may say that the "border texture" is different from the textures on either side. We can use this fact to force all the events in the border into a set. Later when this set is colored, the border will appear distinctly.

Let t_1 and t_0 be the protosamples of textures present in a combined scene T^1 . Let us consider the union of t_1 and t_0 as T^0 . We proceed as before and obtain F^1 , F^ϕ , and F^0 . It is intuitively clear that F^0 is a null set because t_1 and t_0 are part of T^1 . F^ϕ contains all events in both T^1 and T^0 . F^1 contains only the events that occur in T^1 exclusively, which are nothing but the border events; therefore, when we color the F^1 set we obtain the location of the border (Figures 5.5 and 5.6).

Figure 5.3a. A Scene with Multiple Textures (t_1, t_2, t_3, t_4)

Figure 5.3b. Binary 'Colored' Image of the Input Scene

(Output of the Filter) with $T^1 = \{t_1, t_2\}$ and $T^0 = \{t_3, t_4\}$

Figure 5.3c. Binary 'Colored' Image of the Input Scene
(Output of the Filter) with $T^1 = \{t_1, t_3\}$ and $T^0 = \{t_2, t_4\}$

Figure 5.4. Filtered Images of Brain Nuclei
 (Codes to extract parameters such
 as maximum diameter, perimeter,
 etc. have been written)

This image shows a blank sheet of handwriting practice paper. It features ten identical rows of horizontal guidelines. Each row is composed of three lines: a top dashed line, a middle solid line, and a bottom solid line. These lines provide a structured framework for practicing letter formation and alignment. The paper is otherwise completely blank, with no text or markings other than the printed lines.

[illegible][illegible]

Figure 5.5. Border Extraction: Example 1

[illegible][illegible]

INPUT PICTURE

FILTERED OUTPUT
 Ø INDICATES TEXTURE BORDER

Figure 5.6. Border Extraction: Example 2

5.3 Iteration

After obtaining $F_c^{1\beta}$ and $F_c^{0\beta}$ for a given pair of textures T^1 and T^0 , if we go back and color the original scenes, we obtain T_c^1 and T_c^0 . This output pair is more easily distinguishable than the original pair. As a matter of fact for an ideal case, T_c^1 and T_c^0 should be uniform and in completely different colors. When this is not the case, as it happens in general, we can repeat the whole process using the colored pair as input pictures.

Let us define an operator C which operates on a pair of scenes:

$$\{T^1, T^0\} \xrightarrow{C} \{T_c^1, T_c^0\}$$

By repeated application of operator C ,

$$\{T^1, T^0\} \xrightarrow{C^n} \{T_{c^n}^1, T_{c^n}^0\}$$

we can go on iterating until we obtain a satisfactory separation in the output pair (Fig. 5.7a-b). The improvement in ROC curve towards the ideal case with each iteration can be seen in Figure 5.7f.

It is clear that each point in the output picture is a function of neighbors around a given point in the original picture. As a result of iteration, the points in the output plane depend (maybe weakly) on a larger and larger area in the original scene. That is to say, we are operating on the original pictures at a more and more global level with each iteration.

The test picture also will be colored iteratively using the results obtained at each iteration and can be subjected to the filtering process with the filters obtained at the last iteration which has given satisfactory separation of the applied pair. At this moment a decision can be made as to which scene the test scene belongs.

5.4 Interval Complexes

In the following examples we have extracted the interval complexes

Figure 5.7a. Textures of Straw (T^1) and Wood Grain (T^0)

Figure 5.7b displays a series of "colored" images of the input pictures at the end of the 5 iterations. The images are arranged in a grid, showing the results of the iterative process. The top row shows the original input images, which are grayscale. The subsequent rows show the results of the iterative process, where the images are colored and the colors are more vibrant and saturated. The images are arranged in a grid, with the original images on the left and the colored images on the right. The colored images show a significant improvement in color quality and contrast compared to the original grayscale images. The iterative process appears to have successfully enhanced the color and contrast of the input pictures, resulting in more visually appealing and informative images.

Figure 5.7b

Figure 5.7b displays a series of "colored" images of the input pictures at the end of the 5 iterations. The images are arranged in a grid, showing the results of the iterative process. The top row shows the original input images, which are grayscale. The subsequent rows show the results of the iterative process, where the images are colored and the colors are more vibrant and saturated. The images are arranged in a grid, with the original images on the left and the colored images on the right. The colored images show a significant improvement in color quality and contrast compared to the original grayscale images. The iterative process appears to have successfully enhanced the color and contrast of the input pictures, resulting in more visually appealing and informative images.

Figure 5.7b. "Colored" Images of the Input Pictures at the End of the 5 Iterations

Figure 5.7c. Same Textures as Shown in Figure 5.7a
Scanned at Coarser Resolution

Figure 5.7d. "Colored" Images of Textures in
Figure 5.7c at the End of 3rd Iteration

• • •

R.O.C (i)

Figure 5.7f. R.O.C. Curves for Both (i) Fine and (ii) Coarse Resolution

with the help of the program written in PL/1 which implements the A^Q algorithm. We obtain the quasi-minimal cover with A^Q algorithm by the disjoint star method.

Example:

8 interval complexes (2-D filters) obtained with herringbone pattern as T^0 and a random picture as T^1 are shown in Figure 5.8a. The multiple texture scene used for analysis is shown in Figure 5.8b. It contains a herringbone texture slightly above the lower righthand corner. Figure 5.8e shows the output of the series of all eight filters for the scene of analysis. As expected, the herringbone texture is blocked. Figures 5.8c and 5.8d show the output of individual filters for the same input scene (Figure 5.8b). It can be seen that features like horizontal lines and vertical lines have been passed through--thus it is demonstrated that some of these filters can be used to extract/enhance some of the features which the human eye can perceive.

Figure 5.8a. Interval Complexes (2-D Filters) for T^1 = Random Texture and T^0 = Herringbone Pattern

Figure 5.8b. Multiple Textural Scene (Input Scene)

Figure 5.8c. Output of One of the Filters with Figure 5.8b as Input

Figure 5.8d. Output of Another Filter for Same Input

Figure 5.8e. Output of All the Filters Placed
in Parallel for the Input Scene Shown in Figure 5.8b

6. FUTURE WORK

6.1 Generalized Linguistic Model

In the literature structural textures are treated differently from those which are mainly statistical in nature. It is our intention to conceive and develop a model adequate to cover both these classes of textures.

In the proposed model, "textures" will be treated as "sentences" generated by a picture grammar. This grammar is characterized by a quadruple $\{V, V_T, P, S\}$ like conventional grammars, where

V_T = terminal symbols

V = alphabet, $V_N \cup V_T$ where V_N = non-terminal symbols

P = production rules

S = starting symbol.

Here the terminal symbols, V_T are pictorial patterns. They may be either "primitives" defined at this level or complex subpatterns developed from primitives in a more refined grammar. The nonterminal symbols can typically consist of intermediate subpatterns, S_J .

The set of production rules are different from those in conventional grammars. Using these rules we must be able to generate a 2-D scene from the "basic patterns," using them as "bricks." The rules should be capable of allowing "growth" of "sentences" in all possible directions (array grammar). Also the production rules can be partly or completely probabilistic.

We have only briefly outlined the minimal requirements of this grammar, which should be able to generate a wide range of textures. It can be easily seen that the "subpattern" can consist of simple elements like a dot, a triangle,

a square etc., or it can have a regular structure like a checkerboard pattern, or it can even be made up of a unit pictorial area with some defined statistical properties. Using one or more of such subpatterns, a textural scene can be generated using either deterministic or probabilistic rules of placement.

Theoretically it appears a feasible model, acting as the "merger" of different models suggested for structural textures and statistical textures separately. We are now studying the practical implications in making it a reality to deal with textures occurring in nature.

6.2 Textural Analysis Using Time Series Analysis

It was our initial intent to make a just choice of parameters needed for our method, like size and shape of the template, etc., based on the information derived directly from the scene of analysis. We resort to time series analysis for a suggestion of the appropriate methodology.

Any digitized visual scene can be represented as a two-way time series. The essential presence of periodicity hints at the possible representation of a digitized textural scene by Seasonal Time Series. This choice seems quite natural and very appropriate because there is a striking correspondence between some of the problems that occur in textures and the problems that are considered in Seasonal Time Series. For example: In textures the repeating subpattern is not necessarily identical from place to place, though it retains similar characteristics. These variations are considered as the presence of white noise, and there are methods to estimate the nature of this noise. As we have seen, there can be more than one periodicity present in textures like the subpattern consisting of sub-subpatterns, and so on. This can be treated as the case of Multiple Seasonality.

Let $Z_1, Z_2, \dots, Z_t$ be a one-way time series.

Let "B" be the "backward operator" such that

$$B Z_t = Z_{t-1}$$

$$B^2 Z_t = Z_{t-2} \text{ and so on.}$$

For non-seasonal series it is usually possible to obtain a parsimonious representation in the form

$$\phi_p(B) \tilde{Z}_t = \theta_q(B) a_t$$

where $\phi_p(B)$ is the "autoregressive" operator, a polynomial in B of order p.

$\theta_q(B)$ is the "moving average" operator, a polynomial in B of order q.

$$\tilde{Z}_t = Z_t - \hat{Z} \quad \hat{Z} = \text{mean of the series.}$$

a_t is the white noise process.

(For further details on this ARMA (autoregressive moving average) and other processes, reference is made to "Time Series Analysis" by Box and Jenkins.)

We extend this concept to seasonal models when we obtain

$$\phi_p(B) \phi_p(B^S) Z_t = \theta_q(B) \theta_q(B^S) a_t$$

where $\phi_p(B^S)$ and $\theta_q(B^S)$ are polynomials in B^S of degrees P and Q, respectively.

"S" is the length of the season. This is a multiplicative model of the order $(p,q) \times (P,Q)$. (For other variations of the same model, refer again to Box and Jenkins.)

This "multiplicative model" is very useful in that it can be easily extended to take care of multiple seasonality. There are methods available to identify these models and estimate the parameters.

So far we have considered only one-way time series. One extension to two-way time series would be to treat it as one-way series by concatenating

either successive rows or columns. By doing this we are imposing one more, of course known, seasonality. Another method would be to perhaps generalize the methods for a 2-D case. Investigations are proposed in this direction.

6.2.1 Interpretation of the Results

We can estimate the "order of dependence" from the scene by time series analysis. That is to say, we can determine the number of nearest neighbors that have influence on the graylevel of a given point. This result will have a strong influence on the choice and shape of the "template". The "length of seasons," of course, gives the idea of the size of the repeating subpatterns and suggests an appropriate choice of "unit cell".

Once we determine the template size from the analysis of "graylevel" time series, we can go one step further to get further results described in the next section.

6.2.2 "Event" Series: Synthesis of Texture

The "Gamma" of an event is described as the unique positive number which is used for identification. It can be looked upon as a linear weighted combination of the graylevels of the neighbors of a point extracted by the template. We can consider "Gammas" of each event extracted at all points as the "event time series". As we do overlapped scanning while extracting events, in this time series dependence of a value as its previous value will be stronger. The order of dependence, which we anticipate in most cases to be Markovian, can be estimated.

From this information we can estimate the probability of any event's occurring in a given scene. Subsequently "likelihood ratios" and so on can thus establish the link between this model and the previous method.

Also we can use this information for synthesizing the textural scene.

We start with an event and the successive choices of the "events" are made based on the above information. Each event can be considered as a "tile" in the same shape as the template and with appropriate graylevels marked on it. We replace each event by these "tiles" and obtain the synthesized textural scene.

6.3 Optimization of Sampling Strategy

We are in a constant search for a scheme by which we can make an optimal choice of parameters that improves the separation of a pair of textural scenes.

6.3.1 Quantization Scheme

When we quantize all the pictures using gray level ranges of equal width, in many cases we might be losing detailed information in larger areas. One way to avoid this is to use "tapered" quantization scheme. Here gray value ranges are adjusted such that in a given experiment each gray level interval occurs with equal probability. The quantization ranges were set once by using a composite gray value histogram derived from all training set samples.

In practical case we found improvement in separation as indicated by R.O.C. curve when we use this quantization scheme.

Investigations are being done to discover such schemes for other parameters like sampling resolution in the initial scan, scanning beam wavelength, etc.

6.4 Multiple Textures

We would like to test the hypotheses that multiple textures will be effectively discriminated by estimating many likelihood ratios, as in the case of multiple signal detection, by experimenting with real textures.

6.5 Analysis of Color Images

So far we have considered only black and white pictures. The spectral (color) information appears to be very valuable in that it is used in many classification schemes for photo interpretation in remote sensing applications and for the interpretation of natural biological images. Multi-spectral discrimination has long been an art based on human judgment and only recently codified for remote sensing applications. We can extend these primitive pattern discrimination procedures to suit the format of our method and so achieve automated analysis of both spatially and chromatically textured scenes.

Like many researchers who attempted this problem, we generate three one-color images (red, blue, and green) from the original color transparency. The triplet of the gray level values of a corresponding point in the three images (R, B, and G) can be treated as an "event". Once we interpret the "event" in this manner, we can proceed in similar lines as before to discriminate among color images.

We can increase the complexity of the "event" by considering the "triplets" of the neighboring points, which are extracted by a template defined. With this we can perform colored texture analysis.

Initial experiments have yielded satisfactory results, and we intend to pursue it further.

Similar work has been done by Young et al, where he makes use of only two quantities (chromaticities), namely $R/R+B+G$ and $G/R+B+G$. The statistics of these properties are used for discrimination.

The above procedure has been suggested to demonstrate the flexibility in the interpretation of events and the subsequent analysis. A set of intervals that can be derived would specify the allowable band of intensities over the spectrum of colors.

APPENDIX

Description of the Programs

Programs have been written for the analysis of texture using signal detection theory techniques in FORTRAN IV. As extensive use of PAX subroutines is made, familiarity with PAX* is assumed in the description of the following programs.

Two main programs have been written. The first one, TEXROC operates very efficiently when the number of events in the universe, E^U is not too large (≤ 4096). The second program NUTEX is designed to handle the case of $E^U > 4096$ but the expected number of distinct events occurring in the scenes of analysis are within the bounds (≤ 4096). Normally the programs run slowly and larger storage area is needed if the upper limit specified above is increased.

TEXROC Program

This program utilizes efficiently all the parallel processing facilities of Illiac III (PAX is a simulator of PAU of Illiac III). The extraction of events from a given picture and the coloring routines are all done in parallel. There are eight entry points for this subroutine:

1. TEXROC	5. COLOR1
2. PICKEV	6. COLOR2
3. ORDER	7. ENDCLR
4. ASNCLR	8. ROC

*See "The PAX II Picture Processing System At the University of Illinois Programming Manual," Edited by R. T. Borovec, Report No. 314, Department of Computer Science, University of Illinois, Urbana, 1969.

Subroutine TEXROC

Purpose:

Initialization routine.

Usage:

CALL TEXROC(KH,KM,NHM,IDL,N1,KF1,N2,KFM,N3,KFO)

Parameters:

- KH, KM - Arrays of length NHM for recording the number of occurrences of each event in the universe in scenes T^1 and T^0 , respectively.
- NHM - Total number of events in the universe.
- IDL - Direction list defining the template.
- KF1, KFM, KFO - Arrays in which events belonging to F^1 , F^0 , and F^0 are stored respectively.
- N1, N2, N3 - Number of events stored in KF1, KFM and KFO respectively. This is set up by the program.

Execution:

Both the arrays KH and KM are cleared and other parameters are passed on for other entry points.

Entry PICKEV

Purpose:

To pick up events from the given picture and store it in an appropriate array.

Usage:

```
CALL PICKEV(IPI,NPI,IPC,KHM)
```

Parameters:

IPI	- Array of plane indices where the picture is stored.
NPI	- Number of planes in stack IPI.
IPC	- Context plane.
KHM	- The array where the number of occurrences of each event is recorded.

Execution:

The stack of input planes is shifted opposite to the directions in the direction list IDL and the resulting pictures are stacked in a new output stack containing $IDL(1) * NPI$ planes. From this new stack the events can be directly read at all points indicated by '1' in the context plane and appropriate locations in the array KHM are incremented.

This subroutine uses $IDL(1) * NPI$ scratch planes. Due to the limitations of PAX, $IDL(1) * NPI$ should not be more than 16.

This subroutine can be called several times to record the events from all the various samples of T^1 and T^0 (which might be from various pictures) in the arrays KH and KM respectively.

Entry ORDER

Purpose:

To partition the set of events into three sets F^1 , F^ϕ , and F^0 and to order the events in F^ϕ according to likelihood ratio.

Usage:

CALL ORDER(KH,KM,NHM,DX,DY,AL,K,Y)

Parameters:

- KH, KM, NHM - Same as before.
- DX, DY - Arrays of length NHM for storing normalized KH and KM arrays.
- AL - Array of length NHM for storing λ of each event. $\lambda = \text{the likelihood ratio} / (1 + \text{likelihood ratio})$.
- X, Y - Arrays for storing coordinates for the ROC.

Execution:

Using the information from the arrays KH and KM, the set of events is partitioned into three sets and stored in the arrays KF1, KFM, and KFO. N1, N2, and N3 are set to the number of events in each of the above arrays respectively. The arrays should be properly dimensioned to accommodate the events.

The likelihood ratio and Lamda is calculated for each event and the KFM array is arranged in descending order according to the likelihood ratio using the SSP subroutine RSRT.

DX, DY and AL arrays are set as follows:

$$\left. \begin{aligned} DX(I) &= KM(I) / \sum_{I=1}^{NHM} KM(I) \\ DY(I) &= KH(I) / \sum_{I=1}^{NH} KH(I) \end{aligned} \right\} \text{ for all } I$$

$$AL(I) = LR(I) / (1 + LR(I)) \quad \text{where } LR(I) = DY(I) / DX(I)$$

Entry ASNCLR

Purpose:

To partition KFM set into two sets using a threshold on lamda.

Usage:

```
CALL ASNCLR(FL,NFLL,KFLL,NFOL,KFOL)
```

Parameters:

- FL - Threshold on lamda
- KFLL, KFOL - Arrays for storing the partitioned sets.
- NFLL, NFOL - These are set to the number of events stored in KFLL and KFOL respectively.

Execution:

The event KFM(I) is stored in KFLL if the corresponding lamda of the event, AL(KFM(I)), is greater than or equal to FL, the threshold specified. Otherwise it is stored in KFOL. NFLL and NFOL return the number of events stored in the above arrays.

Entry PRINTO

Purpose:

To print the arrays.

Usage:

```
CALL PRINTO(NF1L,KF1L,NFOL,KFOL,JF1FO)
```

Parameters:

- KF1L, KFOL - Arrays containing the partitioned sets of KFM for any given lamda (FL).
- NF1L, NFOL - Number of elements in the above arrays.
- JF1FO - F1 and F0 arrays are also printed if JF1FO = 1. If JF1FO = 0, it is suppressed.

Entry ROC

Purpose:

To draw the ROC curve on the Calcomp plotter.

Usage:

```
CALL ROC(X,Y,NT,NFLL,JAXES)
```

Parameters:

- X, Y - Arrays containing coordinates of the ROC curve.
- NT - Total number of points in the ROC curve (N2 + 2).
- NFLL - Number of events in KFLL.
- JAXES - X, Y axes are drawn, with appropriate marking and titles if JAXES = 1, otherwise it will be suppressed.

Execution:

ROC curve is drawn for the X_1 and Y coordinates calculated before. Marks are made at the end of F^1 , $F^{1\lambda}$ and $F^{0\lambda}$ sets. Multiple curves can be drawn on the same axes by setting JAXES = 0 after the first curve.

Specify Calcomp = Yes on the ID card. The EXEC card should be changed as follows:

```
// EXEC PAX, GOPGM = CALCOMP, REGION.GO = XXXX
```

Entry COLOR1, COLOR2, ENDCLR

Purpose:

To mark (color) the given sets of events of the input picture by the specified greylevel in the output picture.

Usage:

```
CALL COLOR1(IPI,NPI,IPC,IPO,NPO,DX,DY,THR)
```

```
CALL COLOR2(LEQCL,NEQCL,KCLR)
```

```
CALL ENDCLR
```

Parameters:

IPI	- Array of indices of input stack.
NPI	- Number of planes in the output stack.
IPC	- Context plane.
IPO	- Array of indices of the output stack.
NPO	- Number of planes in the output stack.
DX, DY	- Arrays containing normalized weights of each event. (*as calculated in the routine ORDER)
THR	- Threshold on the weights of events. An event, I, is considered for coloring only if $DX(I) + DY(I) \geq THR$.
LEQCL	- An array containing events of an equivalence class.
NEQCL	- Number of events in LEQCL.
KCLR	- The grey level with which the events in LEQCL appears in the output stack. $KCLR \leq 2^{NPO}$.

Execution:

COLOR1 is called to initialize the coloring routines. COLOR2 is called once for each equivalence class and at the end ENDCLR is called. This subroutine uses $IDL(1) * NPI$ scratch planes.

NUTEX Program

This program is a modified version of TEXROC to handle the situation when the number of events in the universe become too large. It needs an estimate of the upper limit on the number of distinct events that can occur in a given experiment. In general we will be dealing with a very small subset of events from the universe. This is not a serious limitation in many cases. If the upper limit is exceeded an error message is printed. Most of the entry points and parameters in this subroutine have the same description as before. Only the differences will be emphasized.

The following entry points have the same description as before:

```
CALL ORDER(KH,KM,NHM,DX,DY,AL,X,Y)
CALL ASNCLR(FL,NFLL,KFLL,NFOL,KFOL)
CALL PRINTO(NFLL,KFLL,NFOL,KFOL,JF1FO)
CALL ROC(X,Y,NT,NFLL,JAXES)
```

Subroutine NUTEX

Purpose:

Initialization routine.

Usage:

CALL NUTEX(IDL,KH,KM,NHM,NMAX,N1,KF1,N2,KFM,N3,KFO)

Parameters:

NMAX - Estimated maximum number of distinct events.

other parameters have the same meaning as before

COMMON/EVENT/KEVENT(NMAX).

This program requires that a common area of length NMAX as shown above be declared in the main program.

Execution:

As error message is printed and the program stops if the number of distinct events exceeds NMAX. $NMAX \leq 4096$.

Entry NUPICK

Purpose:

To extract events in a given window from an input picture and store them in the array specified.

Usage:

CALL NUPICK(IPI,NPI,IWIN,KHM)

Parameters:

IPI	- Array of indices of input planes.
NPI	- Number of planes in IPI.
IWIN	- The window on the input plane from which the events are extracted.
KHM	- KH or KM array depending on whether the scene is T^1 or T^0 .

Execution:

Here the events are extracted in a given window in a serial fashion unlike the routine PICKEV.

Entry DATAAQ

Purpose:

To file the data for the AQ program.

Usage:

```
CALL DATAAQ(THR,KFLE,KFOE,IREV)
```

Parameters:

THR - Threshold on the weight of each event.

KFLE, KFOE - Arrays to store the 'true' and 'false' sets of events.

IREV - If IREV = 1, another data set is filed with true and false sets reversed. (This is the same as reversing the roles of T^1 and T^0 for AQ.) If IREV = 0, only one data set is filed.

Execution:

An event, I, is considered for filing if $DX(I) + DY(I) \geq THR$.
The following information precedes the data:

```
QLQT = '1'B, SAVEC = '1'B;
NV, NL, N11, NOO
KFLE and KFOE
```

77
BIBLIOGRAPHY

1. GENERAL

- 1.1 Chow, C. K. "An Optimum Character Recognition System Using Decision Functions." IRE Trans. on Electronic Computers, EC-6 (December, 1957), 247-54.
- 1.2 _____. "A Recognition Method Using Neighbor Dependence." IRE Trans. on Electronic Computers, EC-11 (1962), 683-90.
- 1.3 Dodwell, P.C. Visual Pattern Recognition. New York: Holt, Rinehart and Winston, 1970.
- 1.4 Evans, D. M. D., ed. Cytology Automation. Edinburgh and London: Livingstone, 1970.
- 1.5 Grasselli, A., ed. Automatic Interpretation and Classification of Images. New York: Academic Press, 1969.
- 1.6 Grusser, O. J. and R. Klinke, eds. Pattern Recognition in Biological and Technical Systems. Berlin: Springer, 1971.
- 1.7 Kaneff, S., ed. Picture Language Machines. London: Academic Press, 1970.
- 1.8 Lipkin, Watt, and Kirsch. "The Analysis, Synthesis and Description of Biological Images." Annual of the New York Academy of Science, 128 (1966), 984-1012.
- 1.9 Narasimhan, R. "Picture Languages." in Picture Language Machines. Edited by Kaneff. London: Academic Press, 1970.
- 1.10 Prewitt and Mendelsohn. "The Analysis of Cell Images." Annual of the New York Academy of Science, 128 (1966), 1035-53.
- 1.11 Rosenfeld, A. Picture Processing by Computer. New York: Academic Press, 1969.
- 1.12 Watanabe, S., ed. Methodologies of Pattern Recognition. New York: Academic Press, 1969.
- 1.13 Watanabe, S. Knowing and Guessing. New York: John Wiley and Sons Inc., 1969.
- 1.14 Wied, G. L. and G. F. Bahr, eds. Automated Cell Identification and Cell Sorting. New York: Academic Press, 1970.
- 1.15 Zusne, L. Visual Perception of Form. New York: Academic Press, 1970.

2. TEXTURES--~~PSYCHOLOGICAL~~ ASPECTS

- 2.1 Flock, H. R. "Optical Texture and Linear Perspective as Stimuli for Slant Perception." Psychology Review, 72 (1965), 505-16.
- 2.2 _____, and A. Moscatelli. "Variables of Surface Texture and Accuracy of Space Perceptions." Percept. Mot. Skills, 19 (1966), 327-34.
- 2.3 Gibson, J. J. "The Reproduction of Visually Perceived Forms." Journal of Experimental Psychology, 12 (1929), 1-39.
- 2.4 _____. The Perception of the Visual World. Boston: Houghton Mifflin, 1950.
- 2.5 _____. "The Perception of Visual Surfaces." American Journal of Psychology, 63 (1950), 367-84.
- 2.6 _____. The Senses Considered as Perceptual System. Boston: Houghton Mifflin, 1966.
- 2.7 Julesz, B. "Visual Pattern Discrimination." IRE Transactions, IT-8 (1962), 86-92.
- 2.8 _____. "Texture and Visual Perception." Scientific American, (February, 1965),
- 2.9 _____. "Cluster Formation at Various Perceptual Levels." Methodologies of Pattern Recognition. Edited by W. Walter Dunn. New York: Academic Press, 1968.
- 2.10 Pickett, R. M. "The Perception of a Visual Texture." Journal of Experimental Psychology, 68 (1964), 13-20.
- 2.11 _____. "Response Latency In a Pattern Perception Situation." ACTA Psychologica, 27 (1967), 160-69.
- 2.12 _____. "The Perception of Random Visual Texture." Models for the Perception of Speech and Visual Form. Edited by W. Walter Dunn. Cambridge: M. I. T. Press, 1967.
- 2.13 _____. Perceiving Visual Texture: A Literature Survey. Aerospace Medical Research Lab, TR-68-12 (1968).
- 2.14 _____. "Visual Analyses of Texture in the Detection and Recognition of Objects." Picture Processing and Psychopictorics. Edited by Lipkin and Rosenfeld. New York & London: Academic Press, 1970.
- 2.15 _____. Application of Texture Perception In the Analysis of Complex Optical Imagery. Aerospace Medical Research Lab, TR-71-81 (1971).

3. TEXTURES--METHODOLOGIES

- 3.1 Brodatz, P. Textures. New York: Dover Publications, 1966.
- 3.2 Bartels, Peter, and G. L. Wied, "Tumor Cell Diagnosis Based On Stochastic Properties of Digitized Images," Proc. 2-D Dig. Signal Proc. Conf., Columbia, Mo. 1971.
- 3.3 Carlucci, Luigia. "A Formal System for Texture Languages." Pattern Recognition, 4 (1972), 53-72.
- 3.4 Conroy, P. Simulation of Texture by Computer Graphics. TR-14. Toronto: Department of Computer Science, University of Toronto, 1969.
- 3.5 Deutsch, E. S. and N. J. Belknap. Texture Descriptions Using Neighborhood Information. TR-184. College Park: Computer Science Center, University of Maryland, 1972.
- 3.6 Haralick, R. M. and J. D. Bissel. Texture Tone Study With Application to Digitized Imagery. (Computer Abstracts AD-724117), Kansas University, December, 1970.
- 3.7 Hawkins, J. K. "Textural Properties For Pattern Recognition." Picture Processing and Psychopictorics. Edited by Lipkin and Rosenfeld. New York: Academic Press, 1970.
- 3.8 IEEE Transactions on Computers, July, 1972. (Special Issue on Two-Dimensional Digital Signal Processing)
- 3.9 Lipkin, B. S. and A. Rosenfeld, eds. Picture Processing and Psychopictorics. New York: Academic Press, New York, 1970.
- 3.10 Muerle, J. L. "Some Thoughts On Texture Discrimination by Computer." Picture Processing and Psychopictorics. Edited by Lipkin and Rosenfeld. New York: Academic Press, 1970.
- 3.11 Puzin, M. "Simulation of Cellular Patterns by Computer Graphics." Unpublished M.Sc. Thesis, University of Toronto, 1969.
- 3.12 Read, J. S. and S. N. Jayaramamurthy. "Automatic Generation of Texture Feature Detectors." IEEE Transactions on Computers (July, 1972),
- 3.13 Rosenfeld, A. "On Models For the Perception of Visual Texture." Models for the Perception of Speech and Logical Form. Edited by Walter Dunn. Cambridge: M.I.T. Press, 1967.
- 3.14 _____, Y. H. Lee, and R. B. Thomas. "Edge and Curve Detection For Texture Discrimination." Picture Processing and Psychopictorics. Edited by Ripkin and Rosenfeld. New York: Academic Press, 1970.
- 3.15 _____, and B. S. Lipkin. "Texture Synthesis." Picture Processing and Psychopictorics. Edited by Lipkin and Rosenfeld. New York: Academic Press, 1970.
- 3.16 _____, and Thurston. "Edge and Curve Detection for Visual Scene Analysis." IEEE Transactions on Computers (May, 1971).

- 3.17 _____, M. Thurston, and Y. H. Lee. "Texture Edge, Spot and Streak Detection." Proceedings of Two Dimensional Digital Signal Processing Conference Columbia, Mo., 1971.
- 3.18 _____, and E. B. Troy. Visual Texture Analysis. (Computer Abstracts, TR 70-116), Computer Science Center, University of Maryland, 1970.
- 3.19 Stoloff, P. H. "Detection and Scaling of Statistical Differences Between Visual Textures." Perception and PsychoPhysics, 6 (1969), 333-36.
- 3.20 Sutton, R. N. and E. L. Hall. "Texture Measures for Automatic Machine Recognition and Classification of Pulmonary Disease." Proceedings of Two Dimensional Digital Signal Processing Conference, Columbia, Mo., 1971.
- 3.21 Swanlund, G. D. "Design Requirements for Texture Measurements." Proceedings of Two Dimensional Digital Signal Processing Conference Columbia, Mo., 1971.
- 3.22 Trout. "The Description of Texture." File No. 832, Department of Computer Science, University of Illinois, Urbana, 1970.
- 3.23 Troy, E. B., E. S. Deutsch, and A. Rosenfeld. Visual Texture Analysis, 3 (February, 1971), TR-145, At-(40-1)-3662.
- 3.24 Two Dimensional Digital Signal Processing Conference. Proceedings. Columbia, Mo., October, 1971.

4. TEXTURES--APPLICATIONS

- 4.1 Darling, E. M., and R. D. Joseph. "Pattern Recognition from Satellite Attitudes." IEEE Transcriptions, SSC-4 (1968), 38-47.
- 4.2 Gerdes, J. W., W. B. Floyd, and G. O. Richards. Automatic Target Recognition Device. AD 478456, January, 1966.
- 4.3 Goldstein, A., and A. Rosenfeld. "Optical Correlation for Terrain Type Discrimination." Photogram. Eng., -30 (1964), 639-46.
- 4.4 Hawkins J. K., and G. T. Elerding. "Image Feature Extraction for Automatic Terrain Classification." Computerized Imaging Techniques, VI (1967), 1-8.
- 4.5 Kaizer. "A Quantification of Textures on Aerial Photographs." Unpublished M.S. thesis, Boston University, 1955.
- 4.6 Katz, Y. H., and W. L. Doyle. "Automatic Pattern Recognition of Meteorological Satellite Cloud Photography." Memo RM-3412, NASA, Rand Corp., Santa Monica, Calif., 1966.
- 4.7 Rosenfeld, A. "Automatic Recognition of Basic Terrain Types from Aerial Photographs." Photogram. Eng., 28 (1962), 115-32.
- 4.8 _____. "Automated Picture Interpretation." Land Evaluation, Macmillan of Australia, (1968), 187-99.
- 4.9 _____. "A Report on the Symposium on Automatic Photo Interpretation." Annual of the New York Academy of Sciences, 157 (1969), 60-64.
- 4.10 Strong, III, J. P. Automatic Cloud Cover Mapping. TR-163/July, 1971, University of Maryland Computer Science Center, College Park, Md.

5. MATHEMATICAL TOOLS

- 5.1 Anderson, T. W. An Introduction to Multivariate Statistical Analysis. John Wiley & Sons, 1958.
- 5.2 Bhattacharya, P. C. "Order of Dependence in a Normally Distributed Two-Way Series." University of Arizona, 1972 (unpublished note).
- 5.3 Box, J. E. P., and G. M. Jenkins. Time Series Analysis. Holden-Day Inc., 1970.
- 5.4 Green and Swets. Signal Detection Theory and Psychophysics. New York: John Wiley and Son Inc., 1966.
- 5.5 Hancock, J. C., and P. A. Wintz. Signal Detection Theory. McGraw-Hill Book Co., 1966.
- 5.6 Hanna, E. J. Multiple Time Series. New York: John Wiley and Sons, Inc., 1970.
- 5.7 Michalski, R. S. A Geometric Model for the Synthesis of Interval Covers. Report 461, Department of Computer Science, University of Illinois, Urbana, 1971.
- 5.8 Michalski, R. S. and B. H. McCormick. "Interval Generalization of Switching Theory," Proceedings of the Third Annual Houston Conference on Comput. Syst. Sci., Houston, Texas, April, 1971. (See also an extended version in Report No. 442, Department of Computer Science, University of Illinois, Urbana, 1971.)
- 5.9 Milgram, D. L. and A. Rosenfeld. Array Automata and Array Grammars. TR-70-141, NONY-5144 (00), Nov. 1970, Computer Science Center, University of Maryland, Maryland.
- 5.10 Pfaltz, J. L. Web Grammars and Picture Description. TR-70-138, GJ-754, Computer Science Center, University of Maryland, Maryland, Sept., 1970.
- 5.11 Robinson, E. A. Multichannel Time Series Analysis with Digital Computer Programs. San Francisco: Holden-Day Publications, 1967.
- 5.12 Rosenfeld, A. Isotonic Grammars, Parallel Grammars, and Picture Grammars. TR-70-111, NONY-5166(00), Computer Science Center, University of Maryland, Maryland, April, 1970.
- 5.13 _____, and D. L. Milgram. Array Automata and Array Grammars--2. TR-171, NONY-5144(00), Computer Science Center, University of Maryland, Maryland, September, 1971.
- 5.14 Smith, A. R. III. "Two-Dimensional Formal Languages and Pattern Recognition by Cellular Automata." Proc. 12 SWAT, 1971 IEEE Conference (Switching and Automatic Theory).

BIBLIOGRAPHIC DATA SHEET		1. Report No. UIUCDCS-R-531	2.	3. Recipient's Accession No.	
4. Title and Subtitle Analysis of Texture				5. Report Date July, 1972	
				6.	
7. Author(s) B. H. McCormick and S. N. Jayaramamurthy				8. Performing Organization Rept. No.	
9. Performing Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, Illinois 61801				10. Project/Task/Work Unit No. Illiac III	
				11. Contract/Grant No. AT(11-1)-2118	
12. Sponsoring Organization Name and Address U. S. Atomic Energy Commission				13. Type of Report & Period Covered	
				14.	
15. Supplementary Notes					
16. Abstracts <p>A method has been presented for performing systematic analysis of visual texture. It has been illustrated that the method is sufficiently versatile to deal with most of the other problems of texture, viz. texture discrimination/recognition, and texture synthesis. This method is based on the principles of signal detection theory. By extending this method and with the application of Interval Covering Theory, it has been shown that texture feature detectors can be automatically generated.</p> <p>Programs which were written to implement this method have been described.</p>					
17. Key Words and Document Analysis. 17a. Descriptors Visual Texture parallel processing pattern recognition picture processing A ^Q algorithm interval covers likelihood ratio 17b. Identifiers/Open-Ended Terms					
17c. COSATI Field/Group					
18. Availability Statement Release Unlimited			19. Security Class (This Report) UNCLASSIFIED		21. No. of Pages 91
			20. Security Class (This Page) UNCLASSIFIED		22. Price