

Computer Science Dissertation Series

**GENERALIZED EULER NUMBER AND RELATED FEATURES FOR
GREY LEVEL IMAGE CHARACTERIZATION**

Jayanta Kumar Dey
M. Tech. (Computer Science)

A Dissertation Thesis
Under the supervision of


Prof. Dr. Bhargab B. Bhattacharya
Advance Computing & Microelectronics Unit

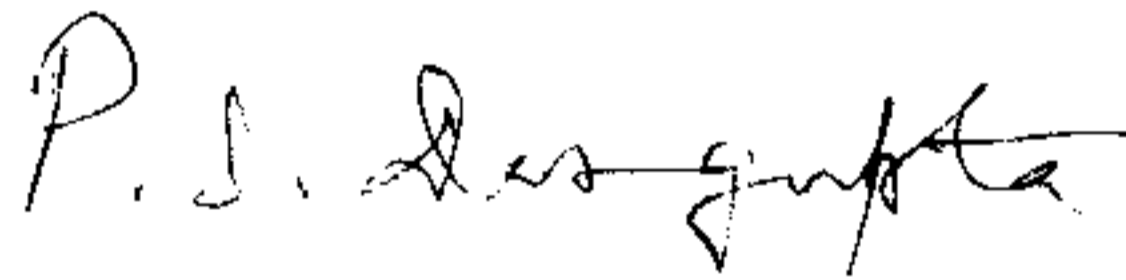
INDIAN STATISTICAL INSTITUTE
203, Barrackpur Trunk Road.
Calcutta – 700 035.
India.

Certificate of Approval

This is to certify that the thesis "Generalized Euler Number and Related Features for Grey Level Image Characterization" submitted by Jayanta Kumar Dey, towards fulfillment of the requirement for the Master of Technology in Computer Science at Indian Statistical Institute, Calcutta, embodies the work done under my supervision during the period from July 1999 to July 2000.

Dated : 27-7-00


Prof. Dr. Bhargab. B. Bhattacharya.
(Supervisor)


(External Examiner)

Acknowledgements

This gives me immense pleasure to express my indebtedness to my supervisor Professor Dr. Bhargab B. Bhattacharya , who despite his hectic schedule always found time to help me with the difficult problems and knew when to trust me with the simpler ones. I also feel eternal gratitude to him for initiating me into the paradigm of mapping of algorithms to architecture.

I am very grateful to Mr. Arijit Bishnu, who in spite of his own hectic schedule always found time to be with me and guided me in each and every step of my dissertation work. Due to his friendly attitude and sincerity to guide me, I never felt fatigued or distressed. I really don't know how to express my gratitude to him.

I would like to thank Dr. Malay K. Kundu of Machine Intelligence Unit, Indian Statistical Institute, who has also supervised me in solving the critical problem of image feature computing. He also provided me with good references from time to time which were very useful to find out the right direction quickly.

I thank Prof. Dr. Bhabani P. Sinha for providing me with the computing facilities of the unit during last one year. He also gave me his valuable suggestion from time to time.

I thank Darpan Majumder for spending his valuable time to discuss with me about my algorithms and their implementation.

I thank the Professors, scientists and staff members of the unit for providing me all the help during the period I worked in the unit.

I must express my heartiest gratitude to my parents. Without their constant inspiration and moral support this work might not have been possible.

Last but not the least, I thank all my classmates and friends who helped me in some way or the other and shared the moments of joy and frustration during my stay at ISI.

Jayanta Dey
Indian Statistical Institute
July 21, 2000.

Abstract

The field of digital image processing is continually evolving. It finds massive applications in areas like medical imaging, remote sensing, geological surveys etc. With the rapid growth in Internet and multimedia technology, the image processing applications are on more demand than ever before. Images and applications built around them involve vast amount of data and therefore takes too much space and time, in other words image processing tasks are computationally expansive. At the same time most of the image processing applications demand real time response.

High performance, special-purpose computer systems are typically used to meet specific application requirements or to off-load computations that are especially taxing to general-purpose computers. With the advancement in VLSI technology hardware cost and size continue to drop. On the other hand processing requirements are becoming well understood in areas such as signal and image processing. As a result systems on VLSI chips are becoming essential to accomplish image processing tasks.

A number of computing features for binary images like Euler number, convex hull etc are present. There are also many fast algorithm present for their implementation. But unfortunately, there is no such measure for grey level images. In our real life, we are much more concerned with grey-level images. Many image processing applications requires recognition of grey level images. In this work I have tried to define few new computing features of grey-level images. I have also described both the parallel and sequential algorithm and their architectural issues.

Contents

Certificate of Approval	3
Acknowledgement	4
Abstract	5
1 Introduction	9
1.1 Background	9
1.2 Why VLSI	9
1.3 Problem Formulation	10
1.4 Organization of the Thesis	10
2 VLSI for Image Processing	11
2.1 Introduction	11
2.2 VLSI Architecture	11
2.2.1 SIMD Arrays	12
2.2.2 MIMD Arrays	12
2.2.3 Pipelined Processors	13
2.3 From Algorithm to Architecture	13
3 Crest Euler Number(CEN) & Valley Euler Number(VEN)	15
3.1 Introduction	15
3.2 Definitions	16
3.3 Examples	18
3.4 Proof of Invariance of CEN & VEN	22
3.5 Computing CEN & VEN	23
3.5.1 Algorithm for Computing CEN & VEN	24
3.5.2 Time Complexity	25
3.6 Experimental Results	26
3.7 VLSI Implementation	28
3.7.1 Parallel Algorithm	28
3.8 Test Images	30
4 Distance Weighted Variance of Grey-Level Value (DWGV)	34
4.1 Introduction	34
4.2 Preliminaries	34
4.3 Computing DWGV	35
4.3.1 Algorithm for Computing DWGV	35
4.4 Experimental Results	36

5	Grey-Level Diffusion Value	38
5.1	Introduction	38
5.2	Definitions	39
5.3	Preliminaries	39
	5.3.1 Mathematical Tools	39
5.4	Computing Grey-Level Diffusion Value	40
	5.4.1 Algorithm	40
5.5	Experimental Results	43
6	Conclusion	45
	Bibliography	46

Chapter 1

Introduction

1.1 Background

Every now and again, our necessity increases. Computers are becoming faster to meet computing requirements, also computing are made faster to grab computer speed. A few years ago, computers were large, room-filling machines requiring ultra clean environment, controlled temperature and humidity, and were to be approached by only authorized staff. These days, "computers" are found in every household. Technical advancement made it possible. Today I am writing my thesis, sitting in my own home, with no extra precaution and arrangement for my PC.

The image processing community have always bemoaned the inevitably heavy demands of their work makes on computer resources. Image contains large amount of data and seem to require highly complex and therefore computationally expansive analytical programs. These demands, coupled with rapidly increasing availability of cheap computer hardware, have led to a proliferation of novel processor architectures. As a result people started implementing various image processing algorithms into hardware architectures. Several work has been reported over past few years on filtering, convolution, geometric warping, cluster analysis, curve detection, radar signal processing, to name a few. However, since most of these systems are built on an ad-hoc basis for specification tasks, methodological work in this area is rare. Because the knowledge gained from individual experiences is neither accumulated nor properly organized, the same errors are repeated. Therefore there is a need for systematic approach for designing algorithm for specific tasks in such a way so that they can be mapped to VLSI architectures automatically or with ease.

1.2 Why VLSI

It is useful to consider why special architectures are needed for image processing. Clearly any algorithm can be implemented on a sequential computer. So why a powerful main-frame or super-computer is not adequate ? The answer is that general purpose computers

can not easily exploit the parallelism in an arbitrary algorithm. Simply improving the raw speed of a sequential computer is not a cost-effective approach for image data which has well defined parallelism. The whole essence of developing special architectures for image processing is to exploit the special forms of parallelism found in image data and algorithms. Specially low level image processing algorithms exhibits higher degree of inherent parallelism and hence most suited for implementation on special architectures.

Nowadays, mature VLSI technology permits the manufacture of circuits whose layout have minimum feature sizes of 1 micron. The effective yield of VLSI fabrication processes make possible the implementation of circuits with order of million transistors at reasonable cost - even for relatively small production quantities. However the advantages of this technology are not fully realized unless simple, regular and modular layouts are used. Fortunately, as mentioned earlier, it has been found that an image processing task can usually be decomposed into a set of sub-tasks distributed over the image. This observations encourage to devise regular and parallel algorithms for image processing tasks which can easily be mapped to regular modules of an architecture.

1.3 Problem Formulation

Image features, specially topological and shape features are very useful for various image processing applications like image characterization, image matching, shape analysis etc. For such applications feature extraction has become an indispensable task. As a case study I tried to develop a content based image retrieval system [9]. I have studied the characteristics of grey-level images. As a conclusion, I found that **there is a trade-off for image characteristics. If we want to get a match between similar images through some image feature, the match will return all similar images, but will also return several dissimilar images. Again, if we want not to get any dissimilar image in return, the matching algorithm will not return all similar images.** Finally we have found three features of grey-level images which are topologically invariant. We have named them (1) CEN-VEN, (2) DWGV (3) Diffusion Value.

We have suggested algorithms for computing these features and tested them with a number of images with different size. All of these measures are parallelly computable. Due to shortage of time we have only implemented parallel algorithm for CEN-VEN measure. These measure is applicable for both grey images and binary images.

1.4 Organization of the thesis.

These thesis is organized in several chapters. In chapter 2 review of earlier work in development of special purpose architectures for image processing tasks has been discussed. Chapter 3 embodies the work on CEN-VEN : definitions, algorithms and results. Chapter 4 and 5 discusses the work on DWGV and Diffusion value. An extensive bibliography given at the end, will help one to find directions for further study and research.

Chapter 2

VLSI for Image Processing

2.1 Introduction

An increasingly accepted contemporary view of image processing is to regard it, in a broad sense, as being nothing more than one aspect of 'information processing' [11]. An 'image' is basically a two dimensional, almost invariably Cartesian, array of data resulting from sampling of the projected instantiation of a local variable, the scene brightness function, obtained through a sensing device, e.g. a camera. The function values are either brightness values or vectors of brightness values sensed in different spectral bands, e.g. color images. The array values are usually integer, non-negative, bounded and implicitly zero outside the field of view bounded by the array dimensions. Image processing algorithms and basic definitions won't be discussed here. There are several good texts which deal with various aspects of image processing in details [7,16,14]. Still for the sake of completeness relevant discussions regarding the shape features will be presented in subsequent chapters.

A large variety of tasks in the field of image processing demand very high rates of instruction throughput. Such high instruction rates can not be supported by conventional serial (Von Neumann) computer architecture. A close analysis of many image processing algorithms reveals that the same sequence of instructions is normally repeated, in an essentially separable manner, on every pixel item, pixel by pixel over the entire image. The inherent parallelism can be exploited to achieve high computational throughput. A number of non-Von Neumann architectures which seem to offer acceptable solution have been exercised and implemented [3]. In the following sections they are discussed.

2.2 VLSI Architectures

Normally two type of parallelism are observed in various tasks and hardware architectures are proposed to use them extensively, spatial and temporal, and image data is no exception. Computer architectures have traditionally been classified according to the uniqueness or multiplicity of their instruction and data streams. By considering the Cartesian product abbreviated as $(SI,MI) \times (SD,MD)$, we obtain four generic

(Single Instruction, Multiple Instruction) X (Single Data, Multiple Data)

architectures : SISD, SIMD, MISD, MIMD [6]. Most conventional uniprocessor architectures are unambiguously classified as SISD, which in effect categorizes the classical Von Neumann architecture. MISD finds no application and use. SIMD and MIMD are discussed below.

2.2.1 SIMD Arrays

SIMD architectures involve a single control unit fetching and decoding instructions, which are then, executed in the control unit itself. Number of processing elements (PEs) are interconnected by a switching network. The PEs operate on their local memories independently. There is no doubt that the SIMD architecture is best fitted to image processing problems as its memory organization matches exactly the image structure and their interconnections between the memory points with their associated processors closely aligns with the neighboring relations between the pixels. As SIMD machines are optimized for parallel neighborhood operations, the data access of each processor is usually restricted to its nearest neighborhood. Typical examples of SIMD architectures are CLIP4 [4], GRID [13], and DAP [5].

2.2.2 MIMD Arrays

An MIMD architecture consists of a number of PEs, each with own program and data. Each PE can communicate with every other one through a communication network. The main feature of MIMD machine is that the overall processing task may be distributed among PEs in order to exploit the inherent concurrency/parallelism in the task and thus decrease the overall execution time. The PEs co-operate in the execution of the overall task by passing messages to each other through communication network.

MIMD architectures can be classified according to their mode of interaction; that is, the degree of coupling and the nature of intercommunication between PEs. Coupling refers to the ability of sharing the system resources. Accordingly MIMD machines may be classified as loosely coupled, moderately coupled and tightly coupled. Loosely coupled systems have a low degree of interaction due to the large geographical dispersion of the system components and low data communication rates. Tightly coupled systems have a high degree of interaction due to close proximity of system components. Moderately coupled systems, also known as distributed computing systems, lies within two extremities of loosely and tightly coupled systems. Image processing tasks requires high speed communication between the PEs and are, therefore, generally implemented on tightly coupled architectures. The location of image(s) in an MIMD system is a significant factor in assigning the viability of the architecture for executing a range of image processing tasks. In an MIMD system the image may be located in the common memory, or distributed among the local memories of PEs. Each PE may have access to the whole image or regions of the image in common memory. Alternatively, each PE may contain a local copy of the whole image or a region of the image. The choice of architectural alternatives depends on the range of image processing tasks to be executed.

There is, therefore a close relationship between machine architecture and image processing algorithm.

MIMD architecture for image processing applications fall into four broad categories : bus architecture, loop architecture, common memory architecture, and reconfigurable architecture. Example of bus architecture is PICAP-II [10]. Example of loop architecture is ZMOB [15]. CYBA-M implements common memory architecture [2]. Computer designers recognized that a machine which is permitted both SIMD and MIMD modes of operation forms an optimal to wide-range of image processing tasks. On top of that mode of operation can be changed or reconfigured dynamically depending upon the application requirements. Example of such machine is PUMPS [1].

2.2.3 Pipeline processors

The concept of pipeline processing has been developed to match an image processing system architecture to serial data inputs. Pipelining of operations is being extensively used in general computing involving operations on vectors and matrices. Typical examples are the Floating Points Systems array processors. Here parallelism is achieved by overlapping several phases needed for instruction, i.e., fetching, decoding, operand address indexing, operand fetching and execution. In the image processing context, pipelining usually implies concatenation of number of operators performing logical and arithmetic operations on small neighborhoods. Typical representative of this category of systems is CYTOCOMPUTER [17].

2.3 From Algorithm to Architecture

Principal goal of this dissertation is to develop architecture to facilitate image feature computation by designing suitable algorithms for specific tasks which exhibits maximum concurrency in execution steps. Designing of algorithms for specific task and developing VLSI architectures apparently seem to be different domain of concentration. Hence is a need for development of a formally sustainable, implementable connection between these two domains. Efficient implementation of complex high performance systems embodying considerable inherent parallelism, as is to clearly appropriate for image processing, will impose some fundamental high level constraints on the system designer. The data structures need to be matched, as far as possible, to the implemented processor-memory-communications architectural topology. High level languages are needed for specification, description and coding. Some effective means of 'concurrency extraction' need to be available, preferably as a tool within the system designing environment. The facility to perform various types of optimization needs to be available so that complex trade-offs can be constructively explored. This eventual goal is to establish an environment where 'from algorithm to architecture' mapping can be done automatically with best possible optimizations exploring various design alternatives. Irrespective of whether we are trying to automate the whole design cycle or executing them step by step, a formal framework is mandatory if a manageable and sustainable design methodology is to be

developed. Theoretical tool, such as complexity theory, are required if appropriate measure of performance and efficiency are to be incorporated into the design path, both at algorithmic and architectural levels.

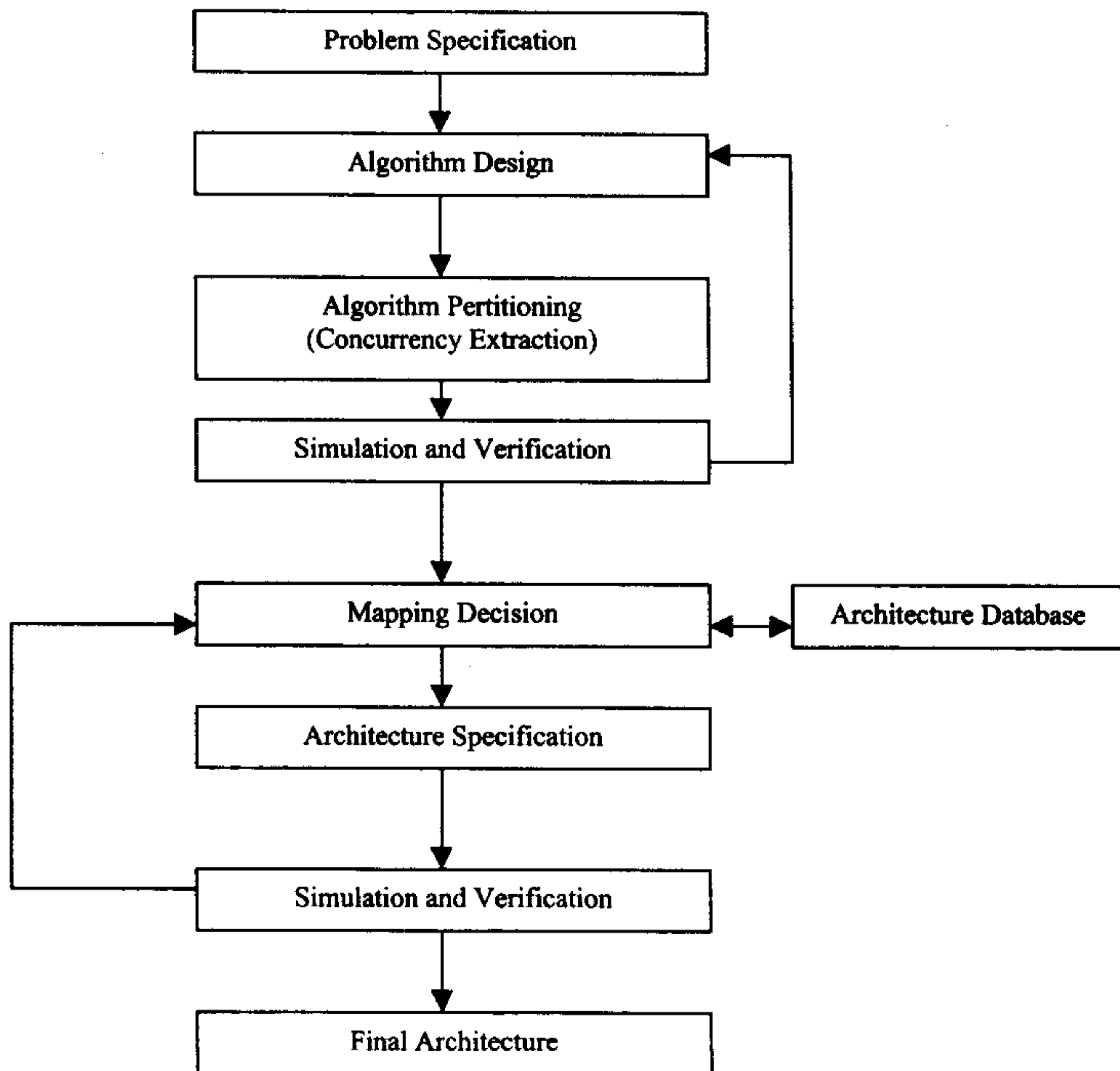


Fig 2.1 : Flow chart of the algorithm to architecture mapping

Chapter 3

Crest Euler Number & Valley Euler Number

3.1 Introduction

This work is very much related with Euler number, which is defined for binary images and that concept is extended to grey level images. It is very useful to discuss a little bit about Euler number, and also how the concept is related to our new measure for grey level images.

The Euler number (or Genus) is the difference between the number of connected components and the number of holes [12, 7]. It is a topological property. Topological properties serve the purpose of geometric shape representation of an image. Topological properties are invariant under any arbitrary "rubber sheet" transformation [7, 8, 14]. Hence they are very useful in image characterization and can be used for matching shapes, recognizing objects, image database retrieval, and many other image processing and computer vision application.

In our real life, it is not sufficient to analyze only binary images. Today many image processing applications require some measure which are defined for grey-level images. It is not always sufficient to use the measures, defined for binary images only, after binarizing the grey level images with a suitable threshold value, because once we binarize a grey-level image it loses many information. Also choice of suitable threshold value also plays important role in it. For example say we want to classify a number of grey-level images into few

in grey-level domain. Only those properties of images which are defined for binary images only will not suffice in these cases. So we are to define and grab some properties of images which are the properties of the grey-level image itself.

Taking the above criteria in consideration we have defined a measure which is solely dependent on the grey-level image itself. Our defined measure for a grey-level image is an ordered pair of "Crest Euler number" and "Valley Euler number". This measure for grey-level images is invariant under "rubber sheet" transformation. It is also invariant under image sharpening, contrast enhancement or reduction unless the image loses its visual similarity. The measure is same for an image with its negated image considering the pair in reversed order. In other words, if two images have the same pairs, after reversing the order of exactly one of the pairs, it is with high probability that one of the images is the negated version of the other.

The proposed measure has been implemented with an $O(M \times N)$ or $O(N^2)$ algorithm (as $O(M) = N$). Recent advances in parallel processing and VLSI technology can be exploited to develop high performance algorithm and architecture to achieve real-time response. In this chapter an algorithm has been proposed which computes this new measure of an $N \times N$ image in $O(N)$ time. Our proposed algorithm can be easily implemented in a special purpose VLSI chip which can serve as a co-processor to the host computer.

3.2 Definitions

Scan path : It is the sequence of pixels along a straight line through which we scan.

Axiom : For a pixel in an image there exists at most four different scan paths through it. (See Fig 3.1 for elucidation).

Proof : For an image pixel with its eight neighbors we can find at most four different straight lines through the pixel. If the pixel location is (i, j) , the four directed lines through it are $R_1((i+1, j-1), (i-1, j+1))$, $R_2((i, j-1), (i, j+1))$, $R_3((i-1, j-1), (i+1, j+1))$, $R_4((i-1, j), (i+1, j))$.

Grad of a pixel along a scan path : Grad of a pixel is defined as the difference between the grey value of the pixel under consideration, and that of the pixel previous to it, in that particular scan path. Grad of a pixel may differ if the scan path differs.

Sign of the grad of a pixel may be defined as "+" if it is positive or "-" if it is negative or "0" if it is zero.

Crest : A pixel is said to have a crest if the ordered pair, consisting of the sign of the grad of the pixel and the sign of the grad of the next pixel along any scan path through the pixel under consideration is an element of the set $\{(+, -), (0, -), (+, 0)\}$.

Valley : A pixel is said to have a valley if the ordered pair, consisting of the sign of the grad of the pixel and the sign of the grad of the next pixel along any scan path through the pixel under consideration is an element of the set $\{(-,+), (0,+), (-,0)\}$.

Note : If there is crest - valley conflict to a pixel , it is declared as crest/valley if more number of crest/valley is detected while scanning through four scan paths.If the same number of crest and valley are reported it is declared as crest.

CREST :

- Connected Crest :**
- (i) If pixel P_i is a crest pixel , it is a connected crest.
 - (ii) If two pixels P_i , P_j are crest pixel and they are in eight-neighborhood of each other, they form a connected crest.
 - (iii) If pixels P_i and P_j form a connected crest and pixels P_j and P_k form a connected crest , pixels P_i , P_j and P_k form a connected crest.

Disconnected Crest : Two crests are said to be disconnected crests iff they are not connected crests.

Crest background pixel : Any pixel other than crest pixel is called **Crest background pixel**.

Crest Hole : Crest holes are 4-Neighborhood connected crest background pixels which are encircled by crest pixels (8-Neighborhood connected).

Crest Euler Number : Crest Euler Number or CEN is defined as the difference between the number of connected crests (CN) and the number of crest holes(CH).
i.e. $CEN = CN - CH$.

Valley :

- Connected Valley :**
- (i) If pixel P_i is a valley pixel , it is a connected valley.
 - (ii) If two pixels P_i , P_j are valley pixel and they are in eight-neighborhood of each other, they form a connected valley.
 - (iii) If pixels P_i and P_j form a connected valley and pixels P_j and P_k form a connected valley , pixels P_i , P_j and P_k form a connected valley.

Disconnected Valley : Two valleys are said to be disconnected valleys iff they are not connected valleys.

Valley background pixel : Any pixel other than valley pixel is called Valley background pixel.

Valley Hole : Valley holes are 4-Neighborhood connected valley background pixels which are encircled by valley pixels (8-Neighborhood connected).

Valley Euler Number : Valley Euler Number or VEN is defined as the difference between the number of connected valleys (VN) and the number of valley holes (VH).
i.e. $VEN = VN - VH$.

3.3 Examples

Consider the following figure (fig 3.1), scan paths are shown by arrowhead lines. Let P_i be the pixel under consideration. P_{i-1} and P_{i+1} be the previous and next pixel along a scan path. Grad of P_i is $G(P_i) - G(P_{i-1})$ along scan path R_j , $j = 1(1)4$ where $G(P_i)$ is the grey-value of pixel P_i .

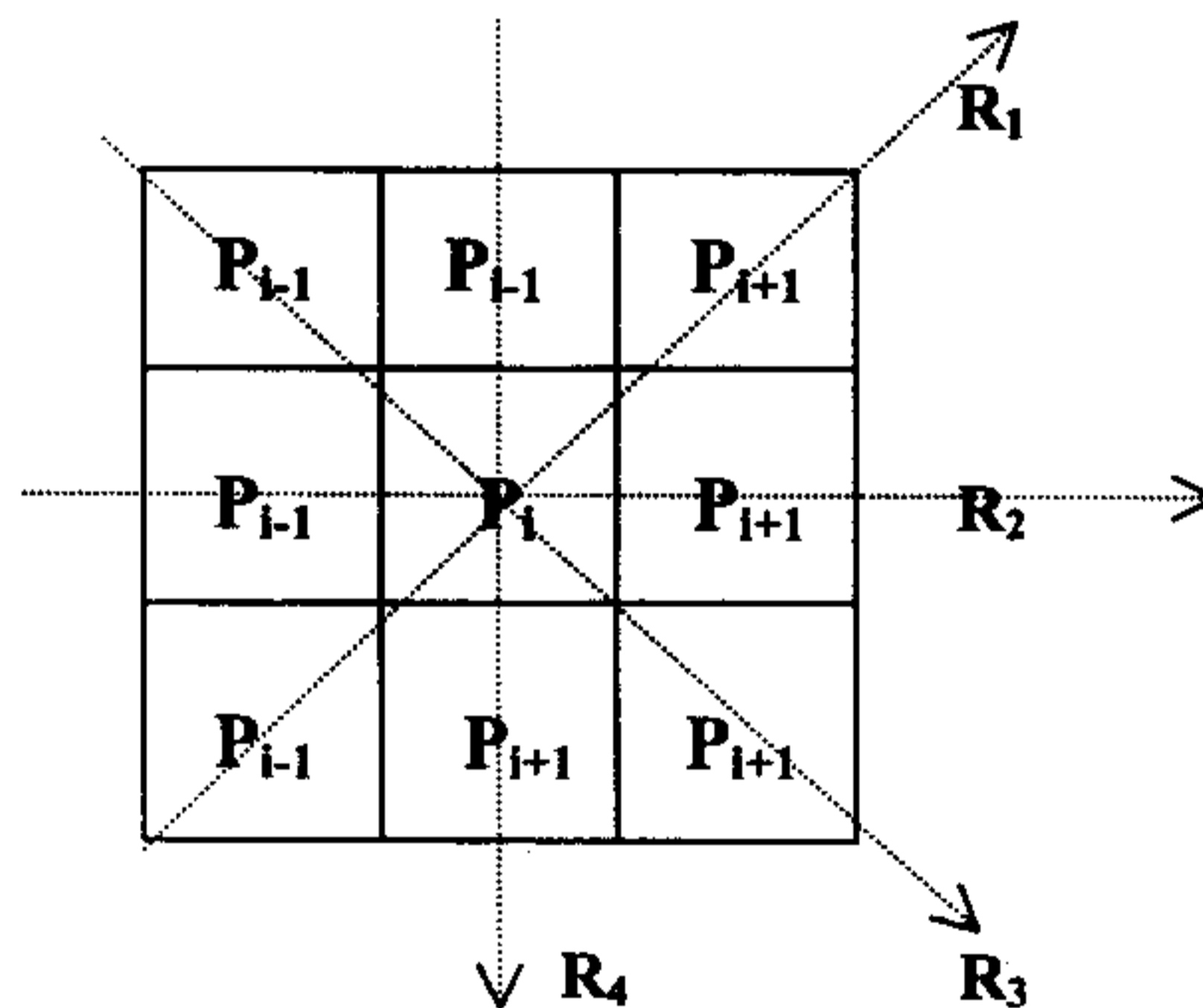


Fig : 3.1_: Labeling of pixels w.r.t. Scan path.

Now let us see figure 3.2. Numbers in square bracket represent the grey value of those pixel. With respect to scan path R_1 the ordered pair, consisting of the sign of the grad of the pixel P_i and the sign of the grad of the next pixel P_{i+1} is (+,-). So there is a crest at pixel P_i .

With respect to scan path R_2 the ordered pair, consisting of the sign of the grad of the pixel P_i and the sign of the grad of the next pixel P_{i+1} is (+,-). So there is a **crest** at pixel P_i .

With respect to scan path R_3 the ordered pair, consisting of the sign of the grad of the pixel P_i and the sign of the grad of the next pixel P_{i+1} is (+,-). So there is a **crest** at pixel P_i .

With respect to scan path R_4 the ordered pair, consisting of the sign of the grad of the pixel P_i and the sign of the grad of the next pixel P_{i+1} is (+,-). So there is a **crest** at pixel P_i .

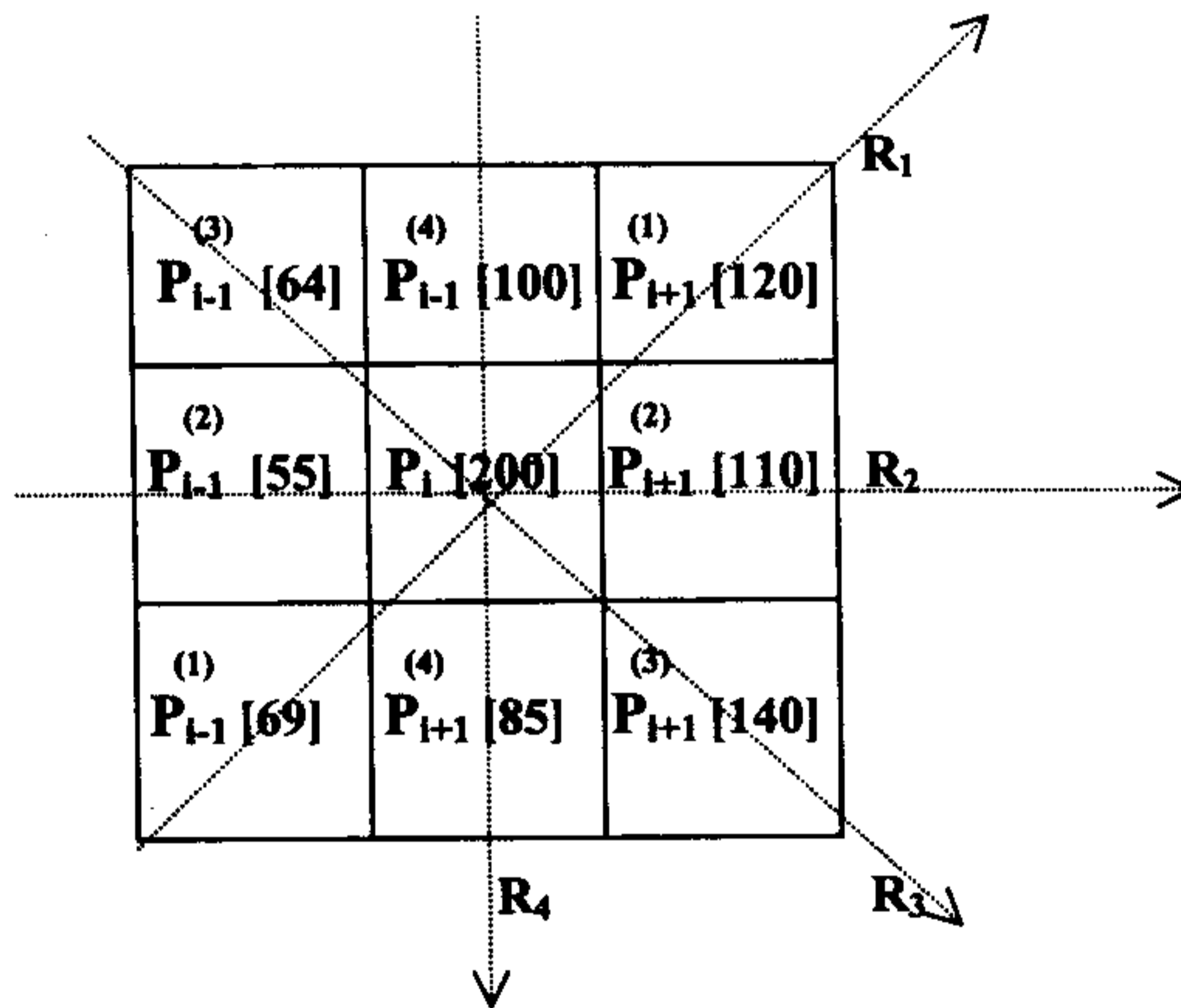


Fig : 3.2 : Labeling of pixels w.r.t. Scan path. Grey value of pixels are shown.

Let us see figure 3.3. Numbers in square bracket represent the grey value of those pixel. With respect to scan path R_1 the ordered pair, consisting of the sign of the grad of the pixel P_i and the sign of the grad of the next pixel P_{i+1} is (-,+). So there is a **valley** at pixel P_i .

With respect to scan path R_2 the ordered pair, consisting of the sign of the grad of the pixel P_i and the sign of the grad of the next pixel P_{i+1} is (-,+). So there is a **valley** at pixel P_i .

With respect to scan path R_3 the ordered pair, consisting of the sign of the grad of the pixel P_i and the sign of the grad of the next pixel P_{i+1} is (-,+). So there is a **valley** at pixel P_i .

With respect to scan path R_4 the ordered pair, consisting of the sign of the grad of the pixel P_i and the sign of the grad of the next pixel P_{i+1} is $(-,+)$. So there is a valley at pixel P_i .

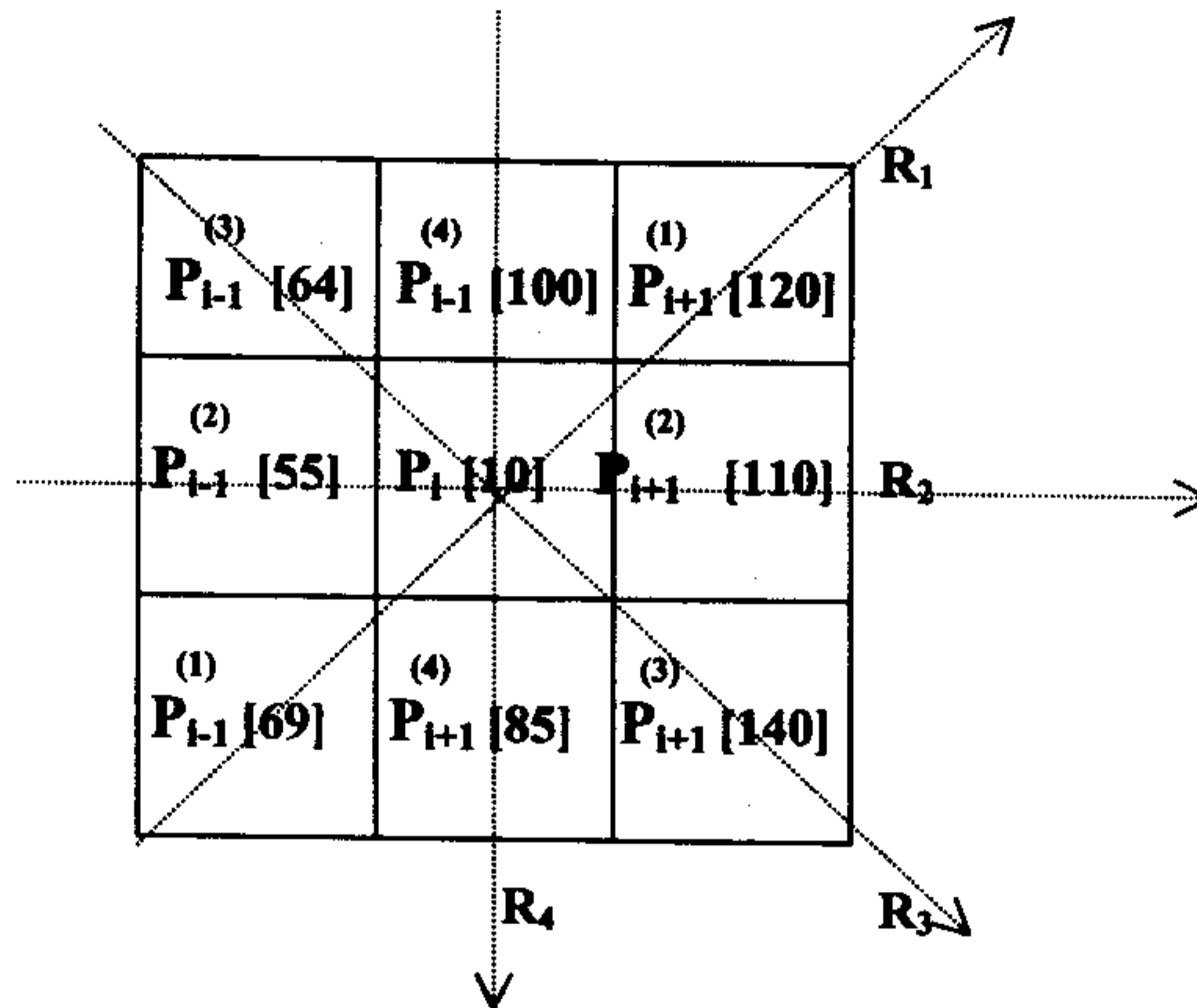


Fig : 3.3 : Labeling of pixels w.r.t. Scan path. Grey value of pixels are shown.

Let us see figure 3.4. Numbers in square bracket represent the grey value of those pixel. With respect to scan path R_1 the ordered pair, consisting of the sign of the grad of the pixel P_i and the sign of the grad of the next pixel P_{i+1} is $(-,+)$. So there is a valley at pixel P_i .

With respect to scan path R_2 the ordered pair, consisting of the sign of the grad of the pixel P_i and the sign of the grad of the next pixel P_{i+1} is $(+,0)$. So there is a crest at pixel P_i .

With respect to scan path R_3 the ordered pair, consisting of the sign of the grad of the pixel P_i and the sign of the grad of the next pixel P_{i+1} is $(+,-)$. So there is a crest at pixel P_i .

With respect to scan path R_4 the ordered pair, consisting of the sign of the grad of the pixel P_i and the sign of the grad of the next pixel P_{i+1} is $(+,-)$. So there is a crest at pixel P_i .

Now as we can see there is a crest - valley conflict at pixel P_i . In this P_i will treated as crest, because in three cases it is declared as crest and in one case it is declared as valley.

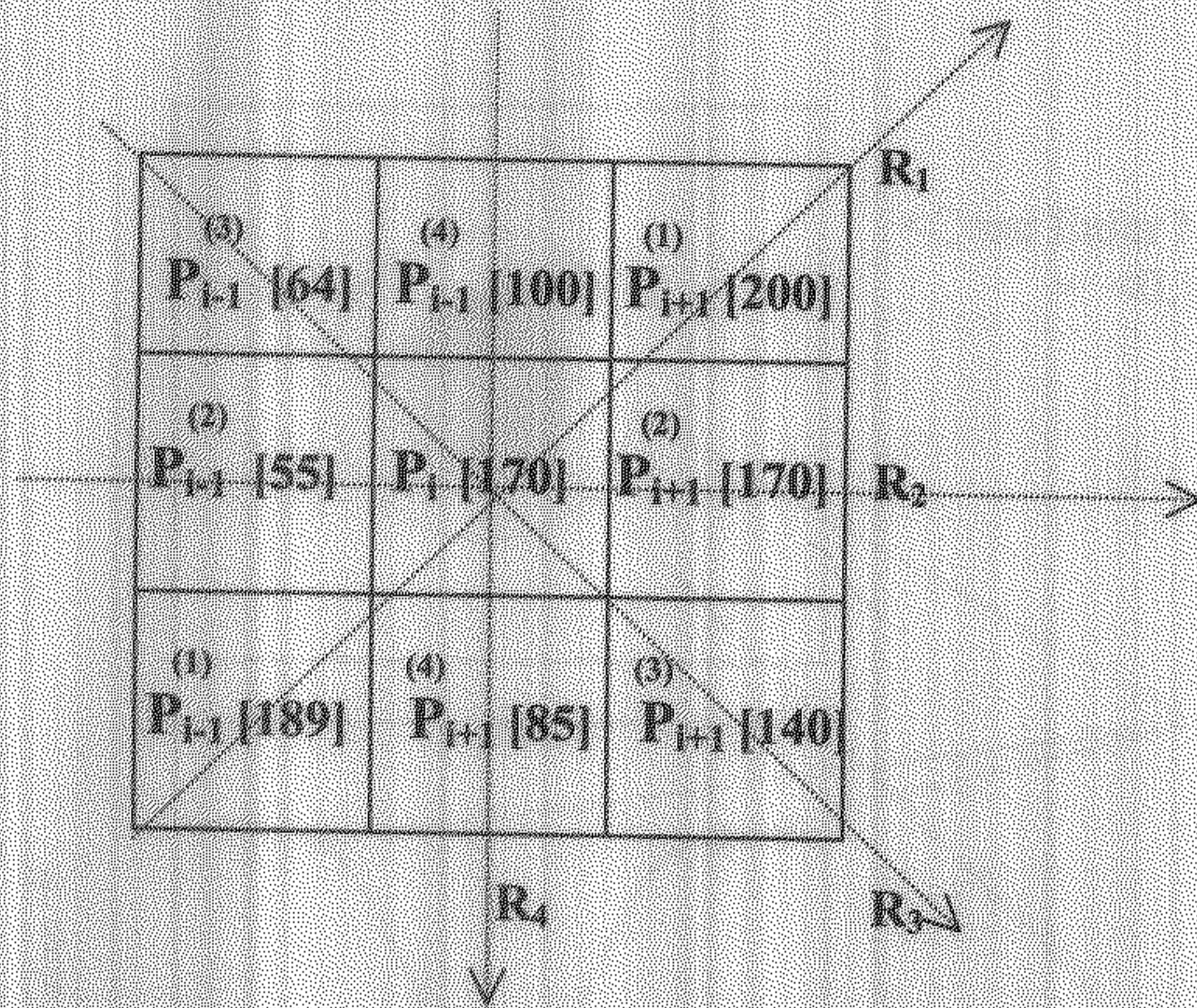


Fig : 3.4 : Labeling of pixels w.r.t. Scan path. Grey value of pixels are shown.

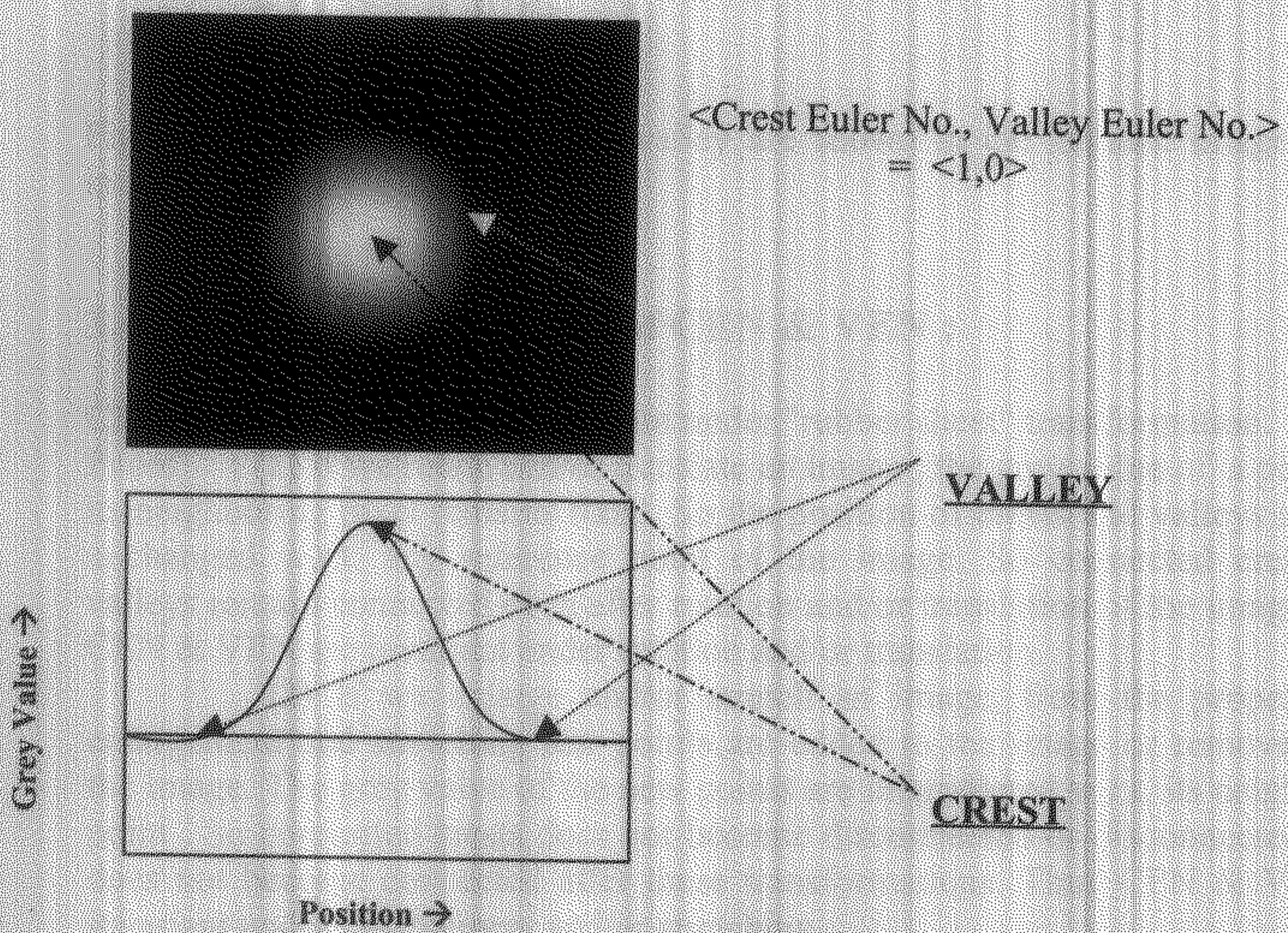


Fig : 3.5 : Position of crest and valley are shown in an image.

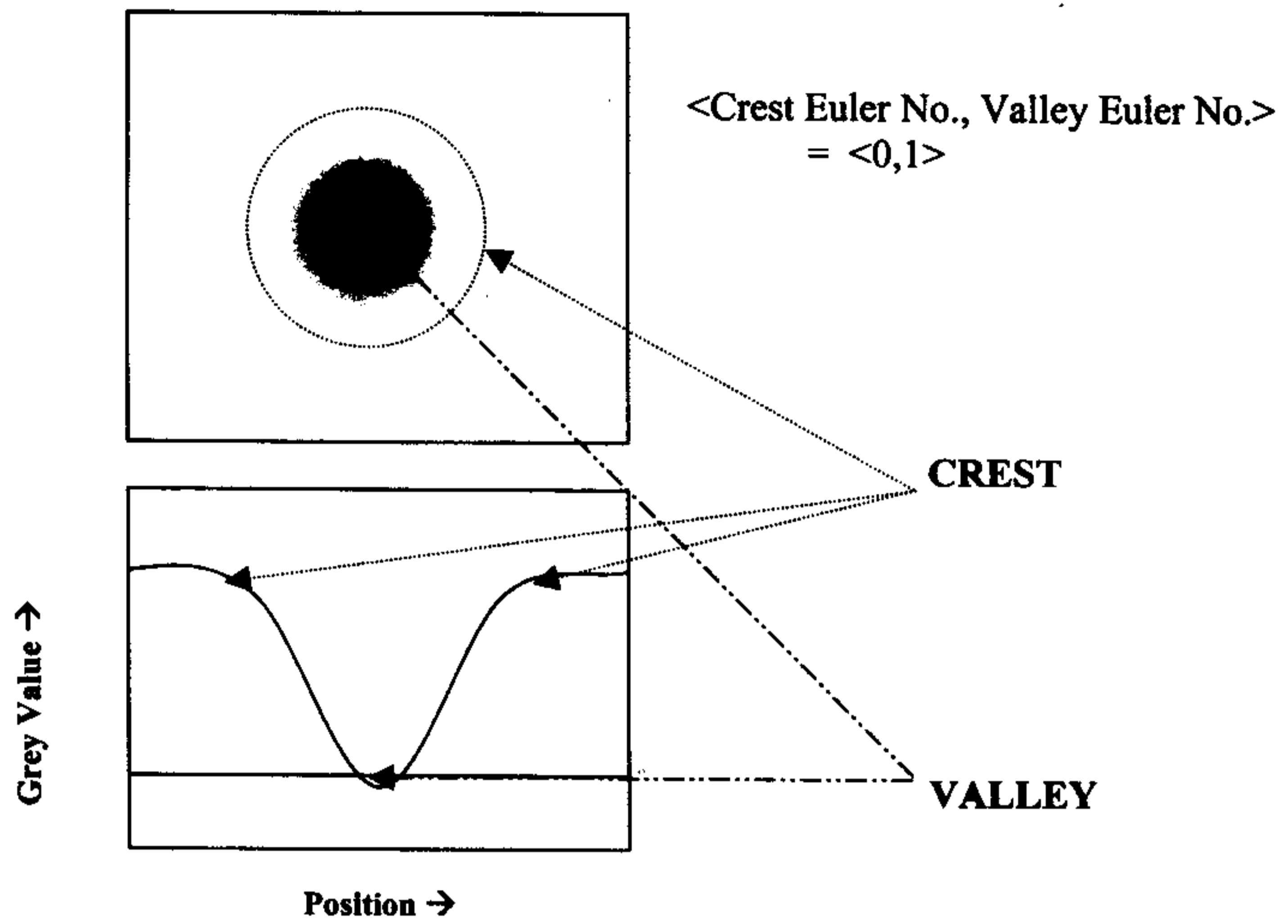


Fig : 3.6 : Position of crest and valley are shown in the negated image of the image shown in fig 3.5.

3.4 Proof of the invariance of CEN & VEN :

Claim 1 : CEN & VEN are invariant under "rubber sheet" transformation.

Proof : (1) If we rotate and/or translate an image, the grey value of a pixel and its neighboring pixel remains fixed. So the position of crests and valleys remains unchanged w.r.t. any reference point of the image. So connected crests and valleys remain connected even after rotation and/or translation of the image. Thus CEN & VEN are invariant under rotation and/or translation.

(2) If we enlarge and/or stretch an image, from the properties of enlargement and stretching few points are added to the image or removed from the image. The grey value of the new points are interpolated w.r.t. its neighboring point. So connected crests does not become disconnected or disconnected crests does not become connected after these transformations. Thus CEN & VEN are invariant under enlargement and/or stretching.

Claim 2 : CEN & VEN are invariant under any filtering applied over it till the image remains visually same w.r.t the original image.

Proof : Whatever filtering is applied over an image, the grey value distribution of a pixel w.r.t. its neighboring pixels remains same, if the resulting image is visually same with the original image, (i.e. if a pixel in the original image has highest/lowest grey value w.r.t. its neighbor's grey value, if, after any filtering applied over the image, the image is visually same with the original image, the pixel under consideration must have highest/lowest grey value w.r.t. that of its neighbors.). So a crest/valley in the transformed image remains unchanged w.r.t. original image. So CEN & VEN are invariant under any filtering applied over it till the image remains visually same w.r.t the original image.

3.5 Computing CEN and VEN

Let S be the input grey-level image matrix. We apply **Mean filtering and Histogram Equalization** over the image S . Histogram equalization tool is available in **MATLAB** toolbox. Let S' be the histogram equalized image matrix. Now according to definitions of **crest** and **valley** we check the locations where crests and valleys exist in S' . Now let T_1 and T_2 be two matrix of same dimension as that of S' with all elements of them initialized to zero (0). Now if there is a crest at (i,j) th location of S' , we put 1 at the same location of T_1 and if there is a valley at (i,j) th location of S' , we put 1 at the same location of T_2 . Now T_1/T_2 are binary images with pixel value "1" for crest/valley and background as "0". Now we compute **Euler Number** for these two binary images.

Theorem : The Euler Number of binary image matrix T_1 gives CEN and Euler Number of image matrix T_2 gives VEN.

Proof : According to our definition CEN/VEN are the difference between the number of connected crest/valley and number of crest/valley holes. Now if pixel P_i is a crest/valley in image matrix S' , it is an object pixel (pixel value 1) in T_1/T_2 and if P_i is a crest/valley hole in image matrix S' , it is a background pixel (pixel value 0) in T_1/T_2 . As we mapped every crests and valleys of image S' in T_1 and T_2 , connected crest/valley remains connected object in matrix T_1/T_2 . As crest/valley holes ($4N$ connected) are encircled by crest/valley pixels ($8N$ connected) in S' , crest/valley holes in S' are mapped to binary image matrix T_1/T_2 as **holes** ($4N$ connected). Euler number is the difference between the number of connected objects and the number of holes. Thus Euler Number of binary image matrix T_1/T_2 gives CEN/VEN of grey-level image S' . From now on we shall call them as CEN/VEN of image S as every image under test will be histogram equalized as pre-processing.

Algorithm 3.5.1 : Computing CEN and VEN of a grey-level image.

Procedure ComputeCENVEN(S, row, col)

**/* S is the input image matrix,
row is the number of rows or vertical length of the image,
col is the number of columns or horizontal length of the image,
*/**

begin

Let T_1 and T_2 be two matrices with same number rows and columns as that of S. Initialize all elements of T_1 and T_2 to 0.

$S' = \text{histogram_equalize}(S);$ **/* S' is Histogram equalize image of S */**

for i := 1 to row-2 do **/* Neglect the edge pixel */**

for j := 1 to col-2 do

 count_cr := count_vy := 0;

**if (((S'[i][j] > S'[i-1][j-1]) and (S'[i][j] > S'[i+1][j+1])) or
 ((S'[i][j] > S'[i-1][j-1]) and (S'[i][j] = S'[i+1][j+1])) or
 ((S'[i][j] = S'[i-1][j-1]) and (S'[i][j] > S'[i+1][j+1]))) **then****
 begin

 count_cr := count_cr + 1;

end

**if (((S'[i][j] > S'[i-1][j]) and (S'[i][j] > S'[i+1][j])) or
 ((S'[i][j] > S'[i-1][j]) and (S'[i][j] = S'[i+1][j])) or
 ((S'[i][j] = S'[i-1][j]) and (S'[i][j] > S'[i+1][j]))) **then****
 begin

 count_cr := count_cr + 1;

end

**if (((S'[i][j] > S'[i][j-1]) and (S'[i][j] > S'[i][j+1])) or
 ((S'[i][j] > S'[i][j-1]) and (S'[i][j] = S'[i][j+1])) or
 ((S'[i][j] = S'[i][j-1]) and (S'[i][j] > S'[i][j+1]))) **then****
 begin

 count_cr := count_cr + 1;

end

**if (((S'[i][j] > S'[i+1][j-1]) and (S'[i][j] > S'[i-1][j+1])) or
 ((S'[i][j] > S'[i+1][j-1]) and (S'[i][j] = S'[i-1][j+1])) or
 ((S'[i][j] = S'[i+1][j-1]) and (S'[i][j] > S'[i-1][j+1]))) **then****
 begin

 count_cr := count_cr + 1;

end

**if (((S'[i][j] < S'[i-1][j-1]) and (S'[i][j] < S'[i+1][j+1])) or
 ((S'[i][j] < S'[i-1][j-1]) and (S'[i][j] = S'[i+1][j+1])) or**


```

        ((S'[i][j] = S'[i-1][j-1]) and (S'[i][j] < S'[i+1][j+1]))      then
    begin
        count_vy := count_vy + 1;
    end
    if (((S'[i][j] < S'[i-1][j]) and (S'[i][j] < S'[i+1][j])) or
        ((S'[i][j] < S'[i-1][j]) and (S'[i][j] = S'[i+1][j])) or
        ((S'[i][j] = S'[i-1][j]) and (S'[i][j] < S'[i+1][j])))      then
    begin
        count_vy := count_vy + 1;
    end
    if (((S'[i][j] < S'[i][j-1]) and (S'[i][j] < S'[i][j+1])) or
        ((S'[i][j] < S'[i][j-1]) and (S'[i][j] = S'[i][j+1])) or
        ((S'[i][j] = S'[i][j-1]) and (S'[i][j] < S'[i][j+1])))      then
    begin
        count_vy := count_vy + 1;
    end
    if (((S'[i][j] < S'[i+1][j-1]) and (S'[i][j] < S'[i-1][j+1])) or
        ((S'[i][j] < S'[i+1][j-1]) and (S'[i][j] = S'[i-1][j+1])) or
        ((S'[i][j] = S'[i+1][j-1]) and (S'[i][j] < S'[i-1][j+1])))  then
    begin
        count_vy := count_vy + 1;
    end
    if (count_vy > count_cr) then
    begin
        T2[i][j] := 1;
    end
    else
    begin
        T1[i][j] := 1;
    end
    endfor
endfor

CEN := CalculateEuler(T1, row,col);
VEN := CalculateEuler(T2, row,col);
return;
end

```

3.5.2 Time Complexity

Asymptotic time complexity of the procedure ComputeCENVEN is $O(N^2)$. To create crest/valley matrix T_1/ T_2 eight access is required for each pixel. So total pixel access is $8N^2$. Procedure CalculateEuler takes kN^2 time. So procedure ComputeCENVEN takes $8N^2 + kN^2$ time which is $O(N^2)$.

3.6 Experimental Result

```
*****  
OUT FILE OBTAINED FROM "file3.c" & "proj3a.c" and over the  
RAW Images (*.H).The idea is to calculate the CREST-EULER  
number and VALLEY-EULER number of the image which are  
defined in the main C file.  
*****
```

```
No of header bytes : 0  
Give the size of image (Row,Column) : 480 640
```

FILENAME	CREST EULER NO.	VALLEY EULER NO.
AFRICA.H	398	1301
ARMY.H	629	1694
BLAZE.H	572	1513
CASTLE.H	1472	3294
CATHED.H	1709	4186
CATTLE.H	1405	4275
CHIMP.H	576	1359
CHOPER.H	1307	3181
COUPLE.H	471	1579
FISH.H	334	709
GOLDFISH.H	304	998
HAWK.H	136	198
ICE.H	104	240
INSECT.H	404	1030
KID1.H	526	895
KID2.H	458	977
KID3.H	198	425
LEAF.H	1231	2368
NEWENG.H	2768	7081
PHOTOGRA.H	1140	3502
RGB3.H	1200	2293
ROCK.H	520	1418
RODEO.H	603	1869
ROSE.H	1757	4308
SANTA.H	262	521
SEAFISH.H	469	1005
SOLDIR.H	2086	4584
STAR.H	1573	2089
SUNSET.H	499	647
TOWN.H	1310	2823
WAVE.H	601	1537
ZEBRA.H	1268	2535

```

/*****/
OUT FILE OBTAINED FROM "file3.c" & "proj3a.c" and over the rotated
RAW Images (*.M).The idea is to calculate the CREST-EULER
number and VALLEY-EULER number of the image which are defined
in the main C file.
/*****/

```

No of header bytes : 0
Give the size of image (Row,Column) : 640 480

FILENAME	CREST EULER NO.	VALLEY EULER NO.
AFRICA.M	398	1302
ARMY.M	629	1694
BLAZE.M	571	1517
CASTLE.M	1471	3295
CATHED.M	1709	4186
CATTLE.M	1405	4275
CHIMP.M	576	1358
CHOPER.M	1306	3178
COUPLE.M	471	1579
FISH.M	334	709
GOLDFISH.M	304	996
HAWK.M	136	198
ICE.M	104	240
INSECT.M	404	1030
KID1.M	526	895
KID2.M	458	977
KID3.M	198	425
LEAF.M	1231	2369
NEWENG.M	2768	7081
PHOTOGRA.M	1140	3502
RGB3.M	1203	2290
ROCK.M	521	1418
RODEO.M	602	1870
ROSE.M	1752	4284
SANTA.M	260	523
SEAFISH.M	471	1001
SOLDIR.M	2081	4583
STAR.M	1573	2089
SUNSET.M	499	647
TOWN.M	1308	2825
WAVE.M	601	1537
ZEBRA.M	1268	2534

3.7 VLSI Implementation

3.7.1 Parallel Algorithm

Algorithm 3.7.1 : *Computing CEN and VEN in parallel.*

Procedure ParallelCENVEN

begin

for each pixel P_{ij} in the image do in parallel

do in parallel

$a = P_{i-1,j+1} - P_{ij}$

$b = P_{i,j+1} - P_{ij}$

$c = P_{i+1,j+1} - P_{ij}$

$d = P_{i+1,j} - P_{ij}$

$e = P_{ij} - P_{i+1,j-1}$

$f = P_{ij} - P_{i,j-1}$

$g = P_{ij} - P_{i-1,j-1}$

$h = P_{ij} - P_{i-1,j}$

enddo

do in parallel

if(((a>0)and(e<0))or((a>0)and(e=0))or((a=0)and(e<0)))

begin

$count_cr_{ij} := count_cr_{ij} + 1;$

end

if(((b>0)and(f<0))or((b>0)and(f=0))or((b=0)and(f<0)))

begin

$count_cr_{ij} := count_cr_{ij} + 1;$

end

if(((c>0)and(g<0))or((c>0)and(g=0))or((c=0)and(g<0)))

begin

$count_cr_{ij} := count_cr_{ij} + 1;$

end

if(((d>0)and(h<0))or((d>0)and(h=0))or((d=0)and(h<0)))

begin

$count_cr_{ij} := count_cr_{ij} + 1;$

end

if(((a<0)and(e>0))or((a<0)and(e=0))or((a=0)and(e>0)))

begin

$count_vy_{ij} := count_vy_{ij} + 1;$

end

if(((b<0)and(f>0))or((b<0)and(f=0))or((b=0)and(f>0)))

begin

$count_vy_{ij} := count_vy_{ij} + 1;$

end

```

if(((c<0)and(g>0))or((c<0)and(g=0))or((c=0)and(g>0)))
  begin
    count_vyij := count_vyij + 1;
  end
if(((d<0)and(h>0))or((d<0)and(h=0))or((d=0)and(h>0)))
  begin
    count_vyij := count_vyij + 1;
  end
enddo

if((count_cr != 0)and(count_vy != 0))
  begin
    if(count_cr >= count_vy)
      begin
        T1(i,j) = 1;
      end
    else
      begin
        T2(i,j) = 1;
      end
    end
  end
endfor

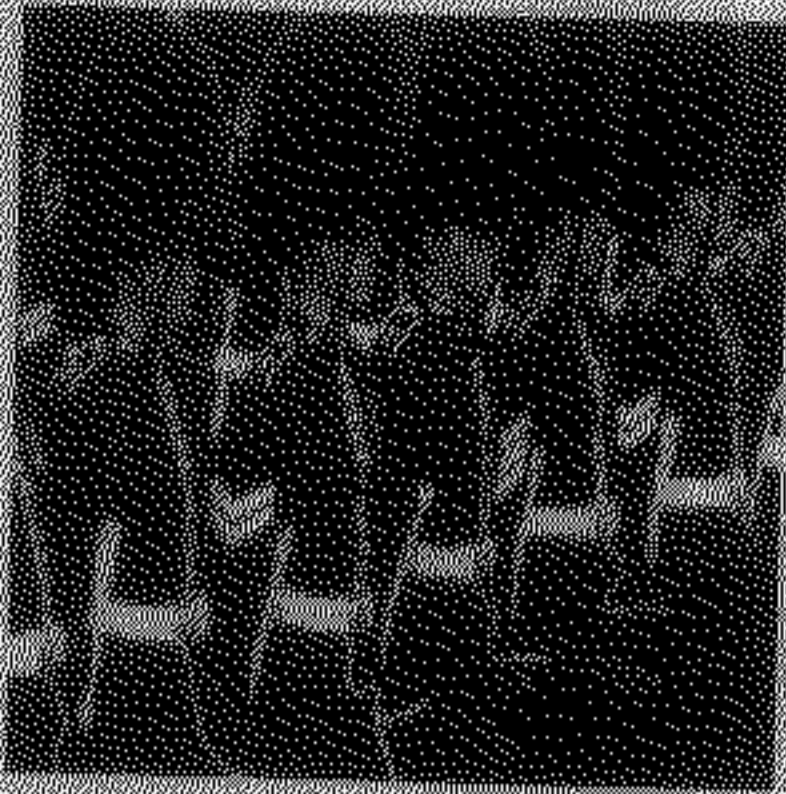
```

Now send matrices T_1 (Crest matrix) and T_2 (Valley matrix) to `ParallelComputeEuler()` to get CEN and VEN.

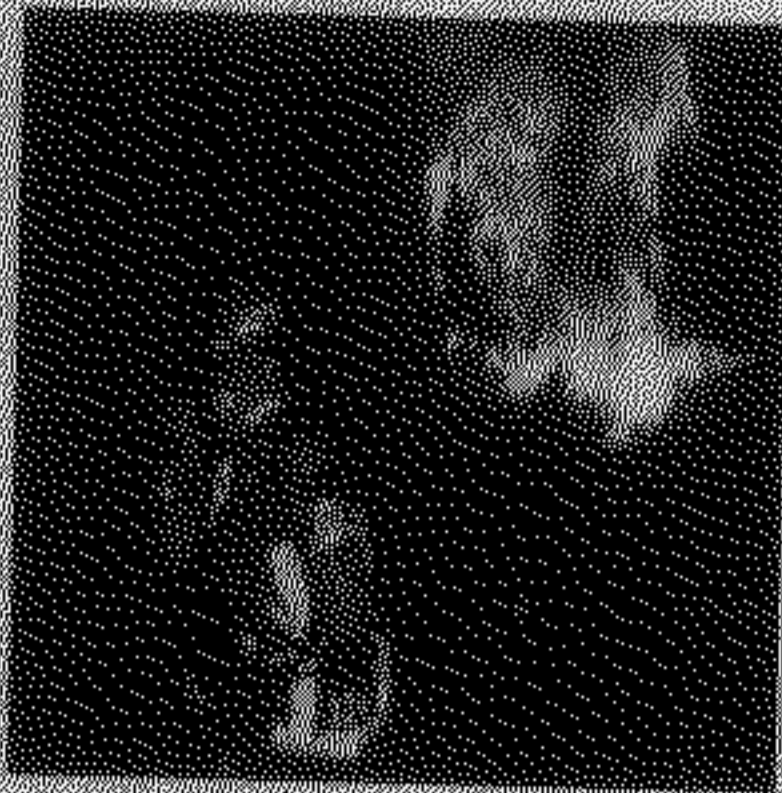
end

3.8 Test Images

SET 1 : Images with actual orientations



ARMY1.JPG



BLAGE1.JPG



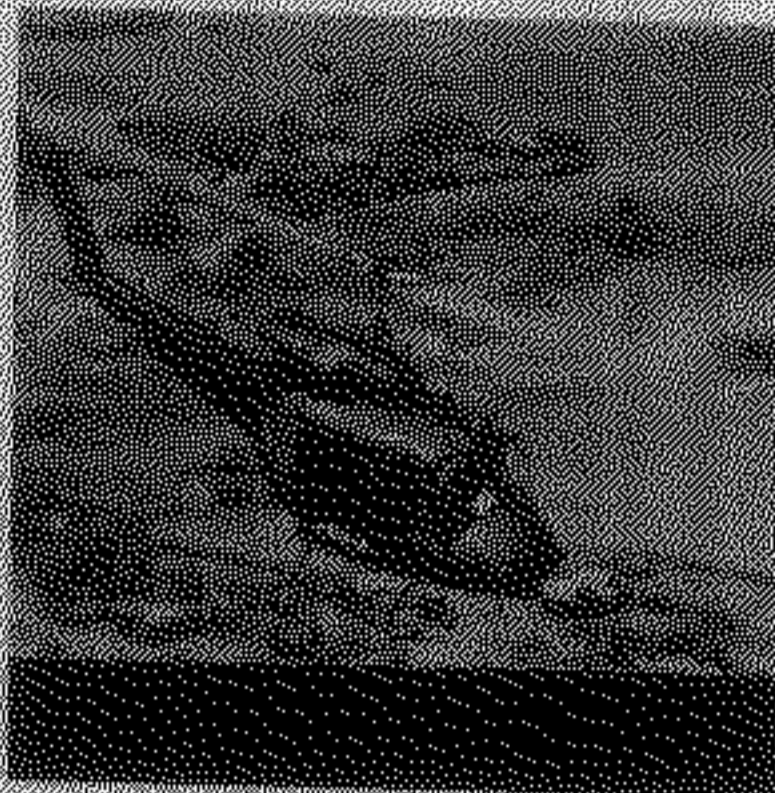
CASTLE1.JPG



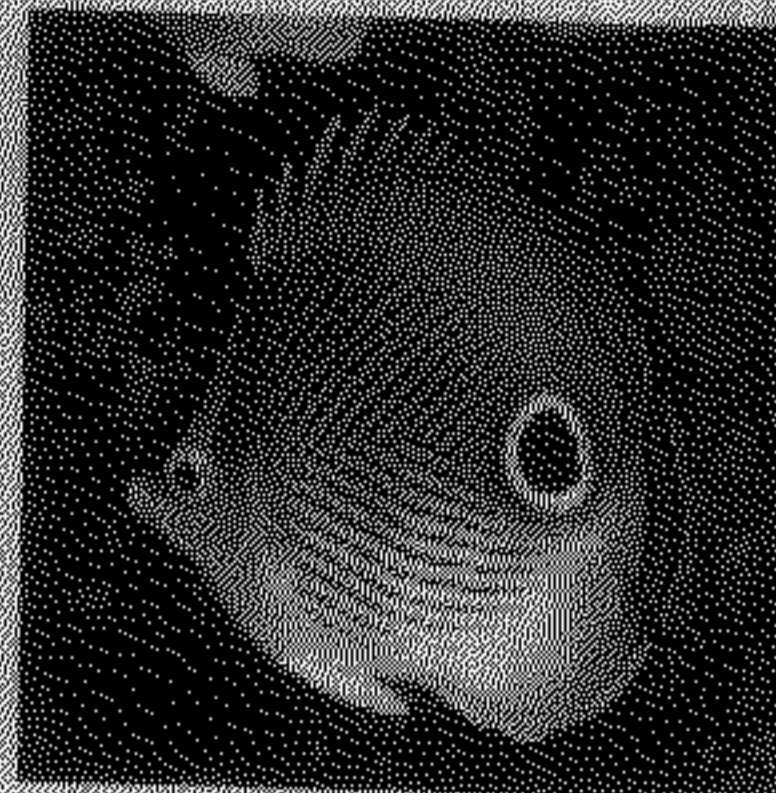
CATHED1.JPG



CHIMPI.JPG



CHOPER1.JPG



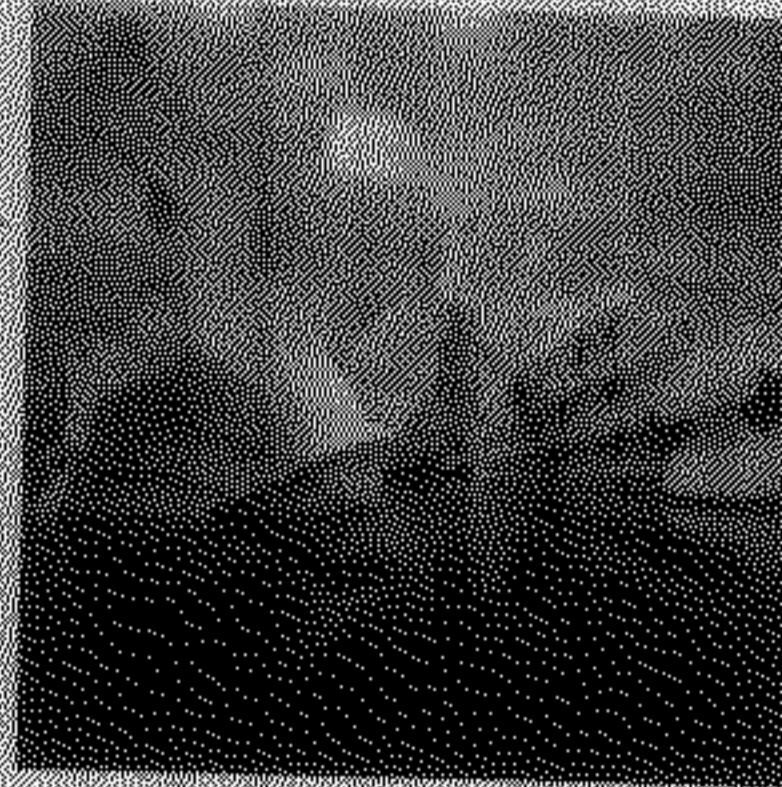
FISH1.JPG



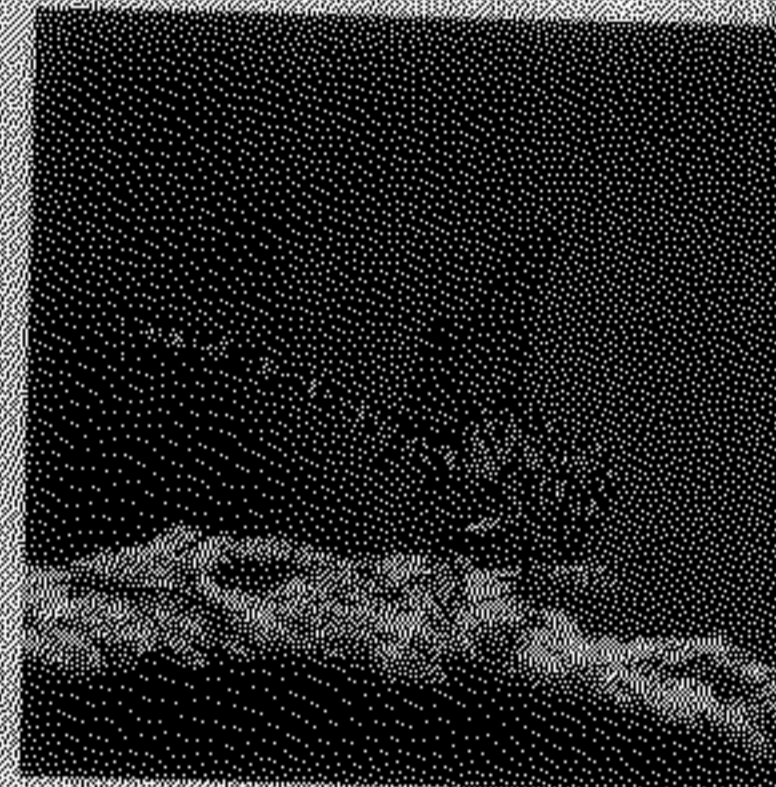
GOLDFISH1.JPG



HAWK1.JPG



ICE1.JPG



INSECT1.JPG



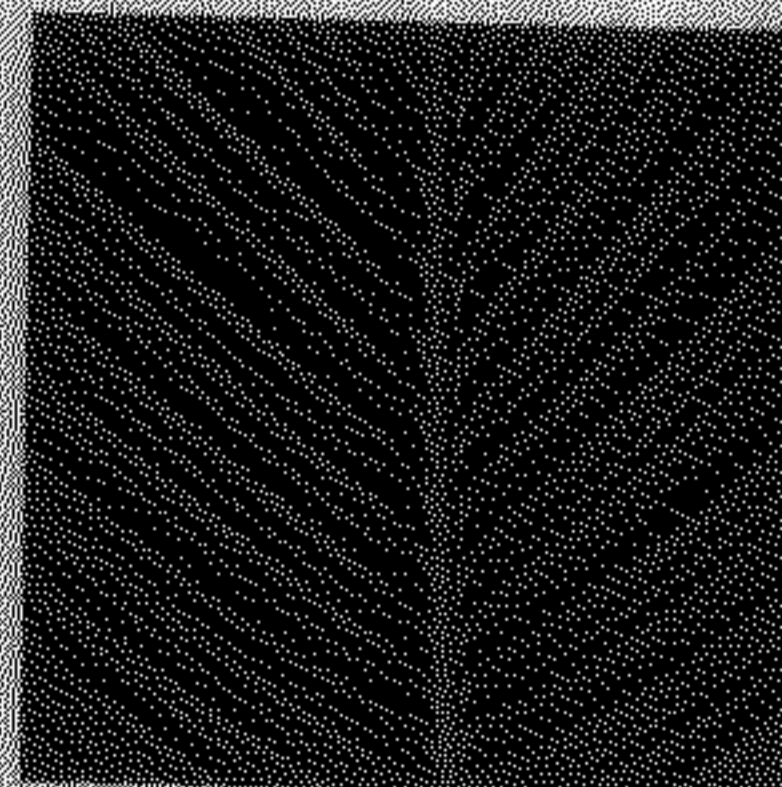
KID11.JPG



KID21.JPG



KID31.JPG



LEAF1.JPG



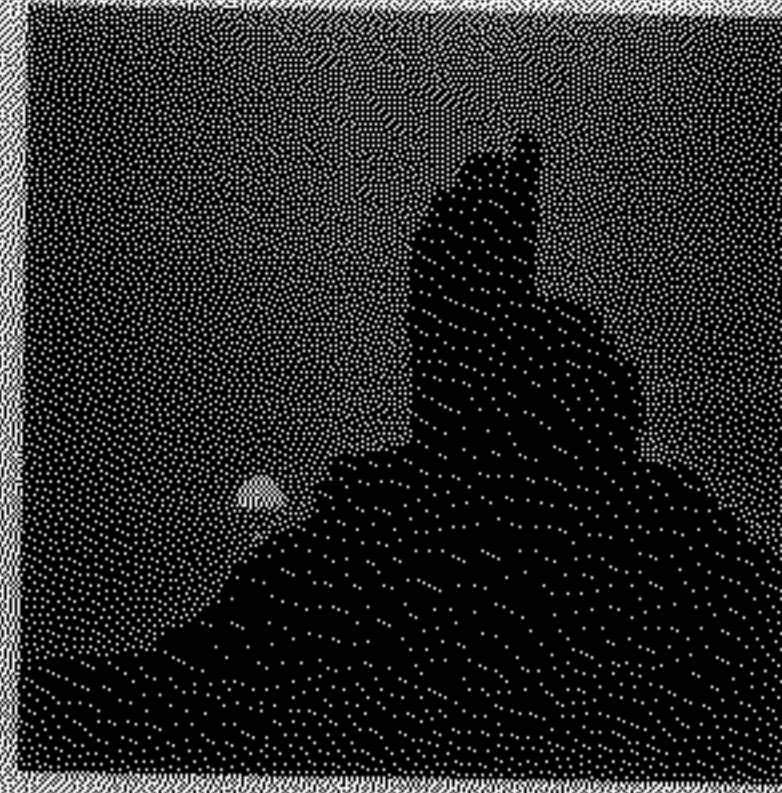
NEWENGL1.JPG



PHOTO1.JPG



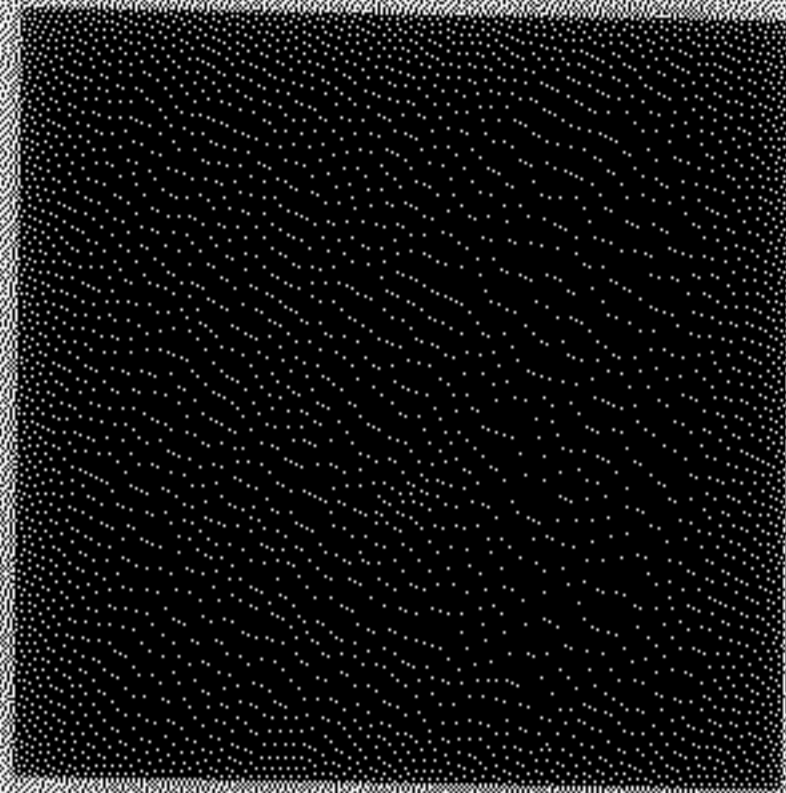
RGB31.JPG



ROCK1.JPG



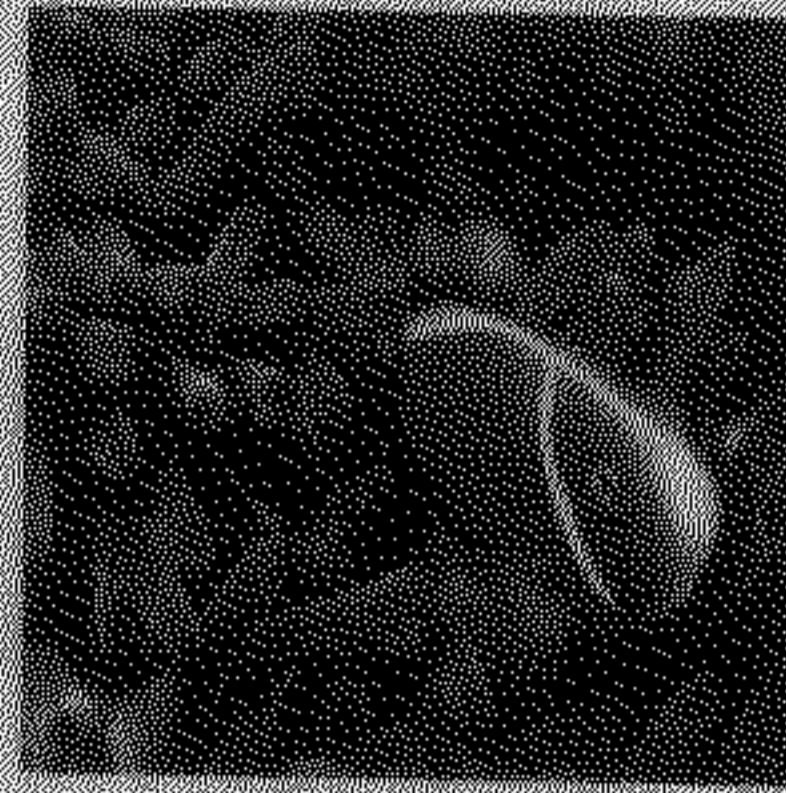
RODEO1.JPG



ROSE1.JPG



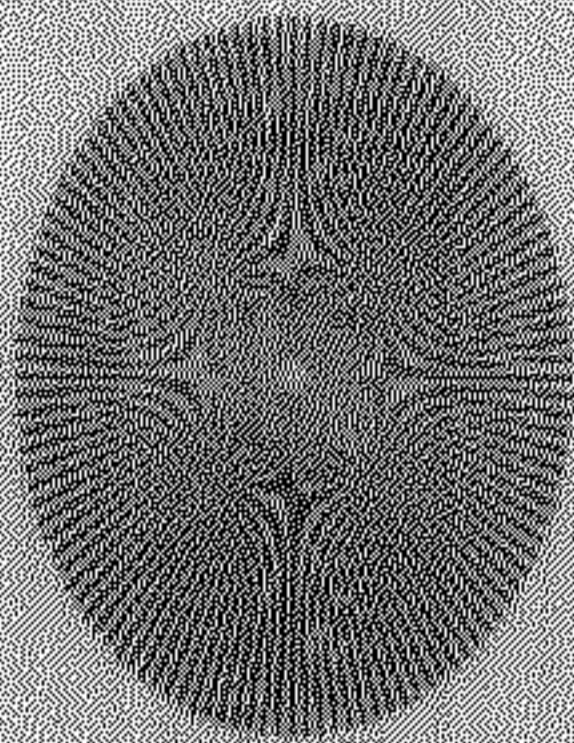
SANTA1.JPG



SEAFISH1.JPG

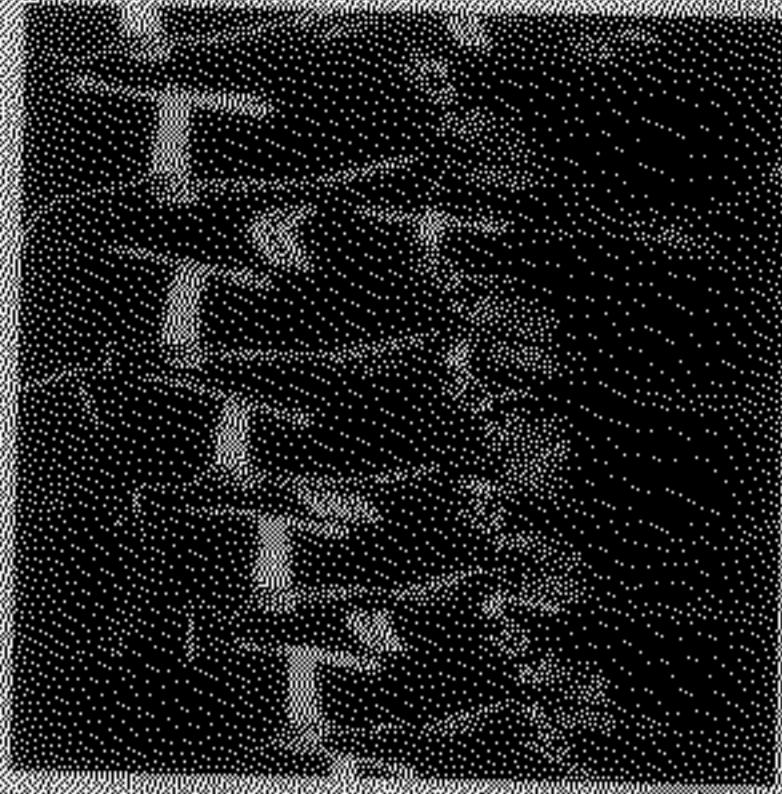


SOLDER1.JPG

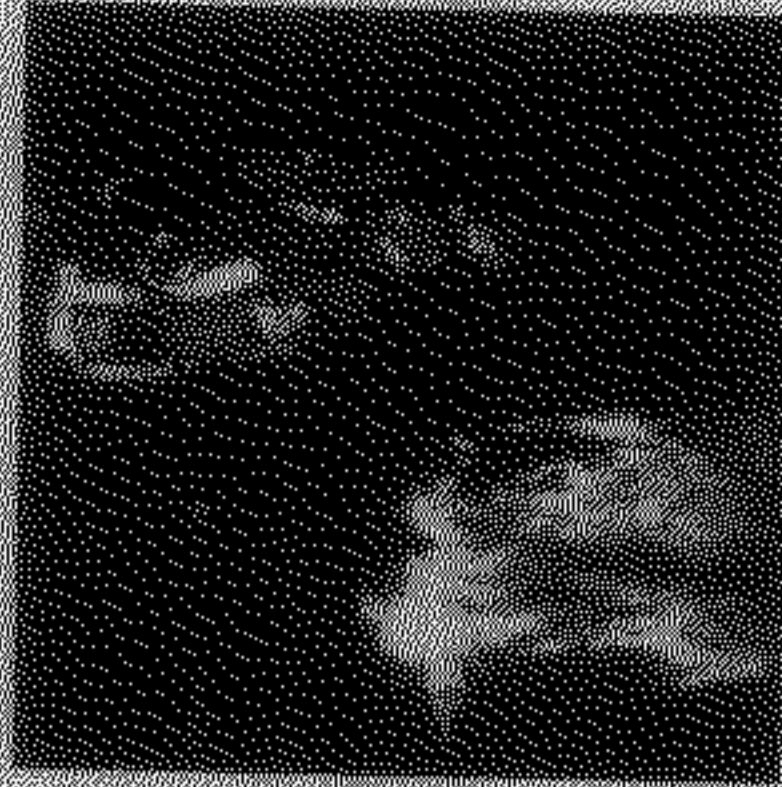


STAR1.JPG

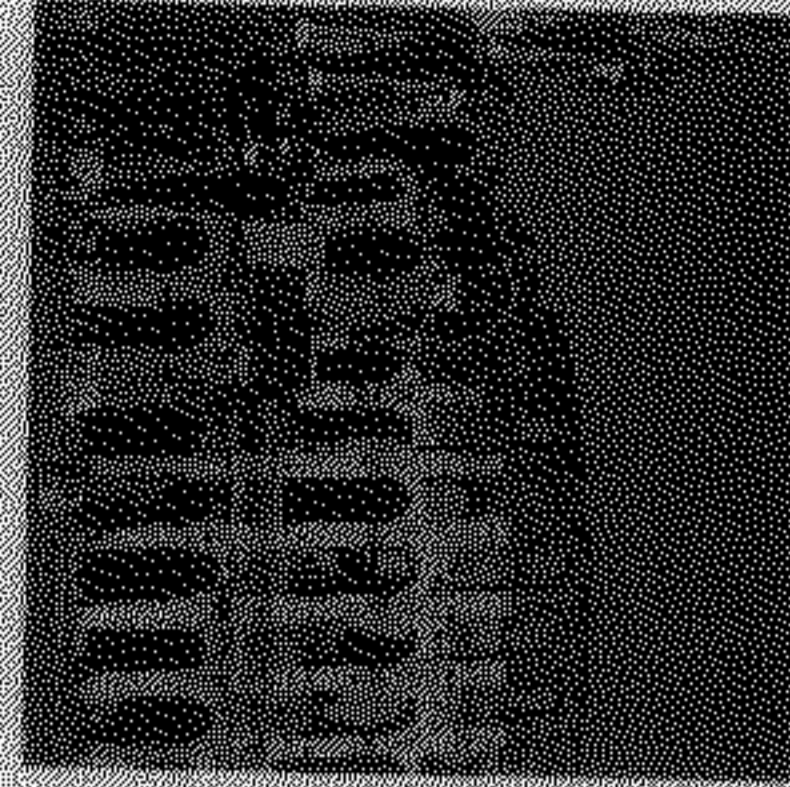
SET 2 : Images with 90° rotation (clockwise) of actual images



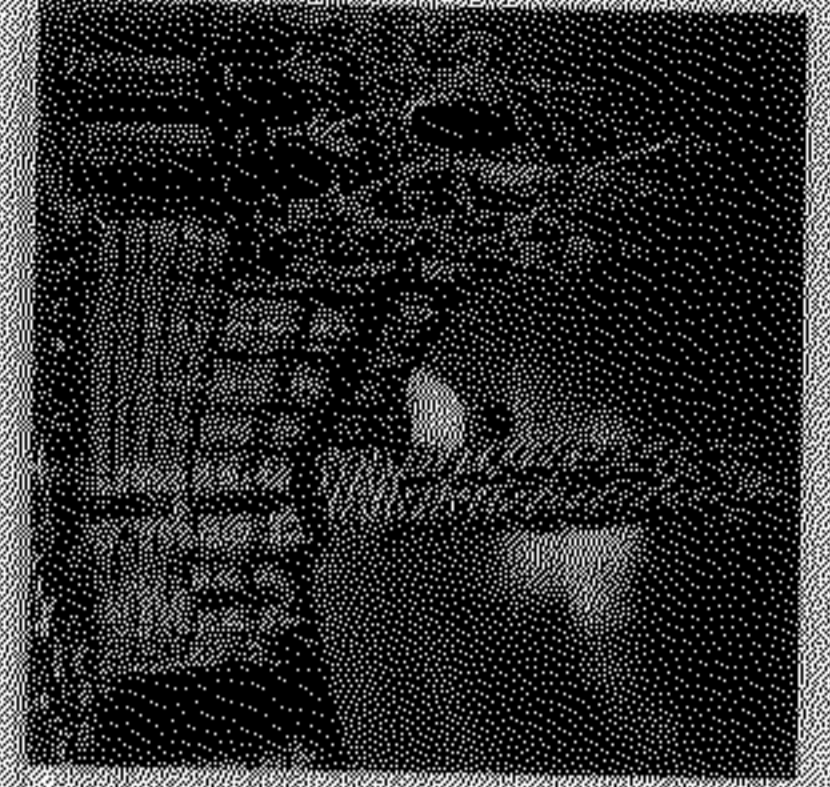
ARMY2.JPG



BLAGE2.JPG



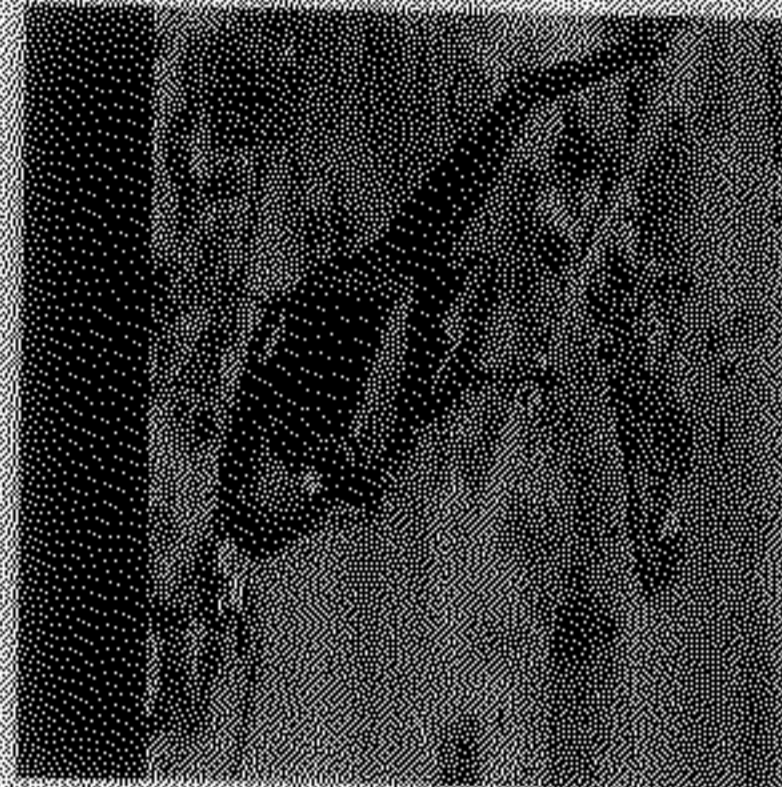
CASTLE2.JPG



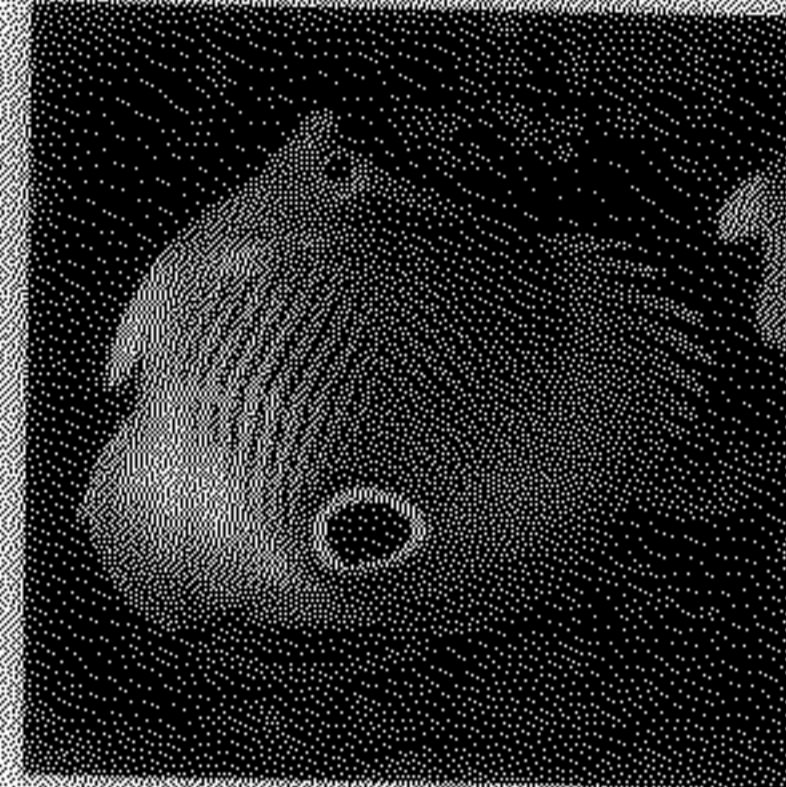
CATHED2.JPG



CHIMP2.JPG



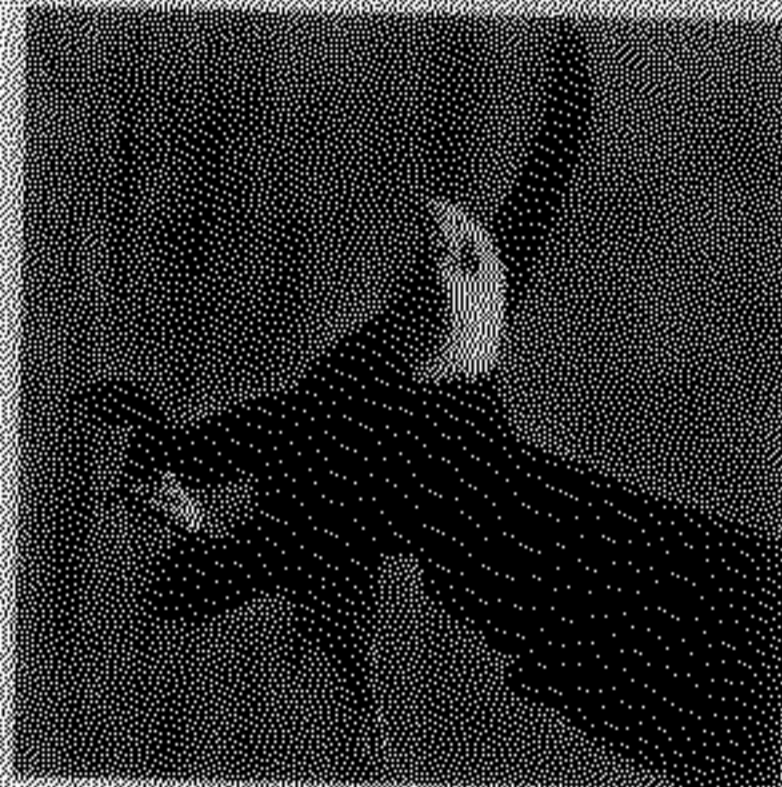
CHOPER2.JPG



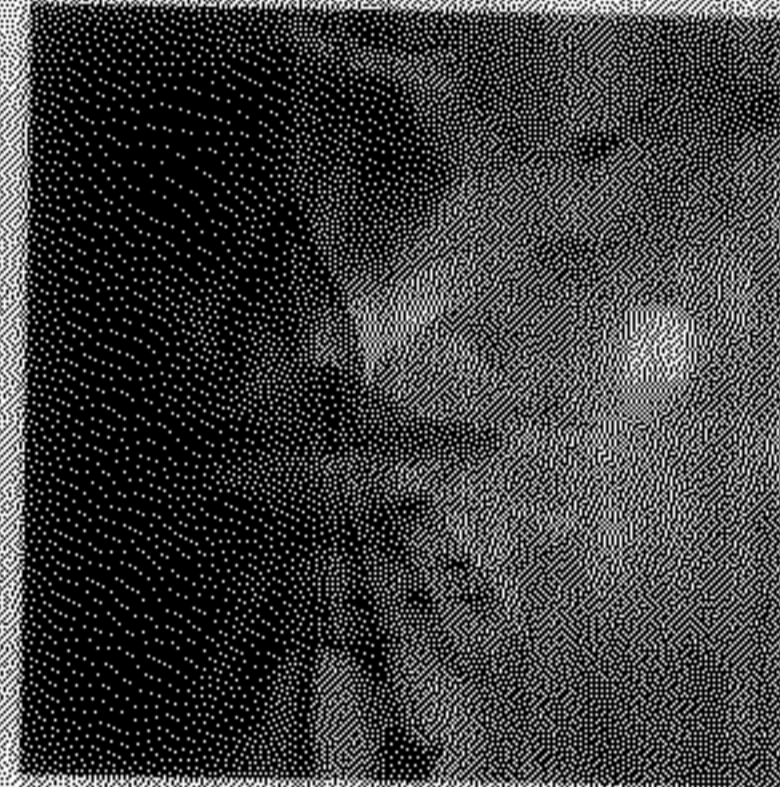
FISH2.JPG



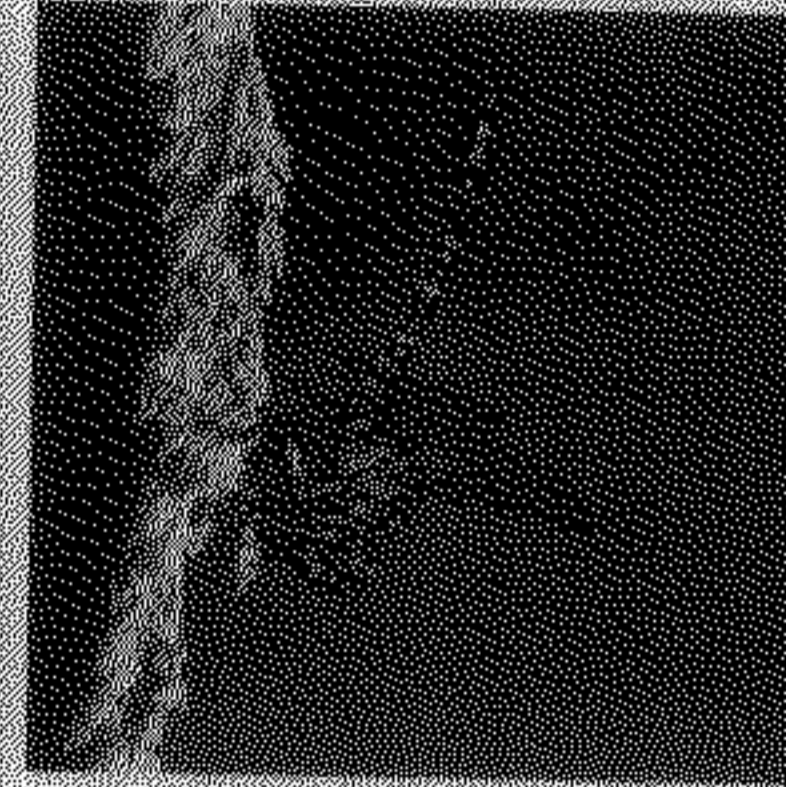
GOLDFISH2.JPG



HAWK2.JPG



ICE2.JPG



INSECT2.JPG



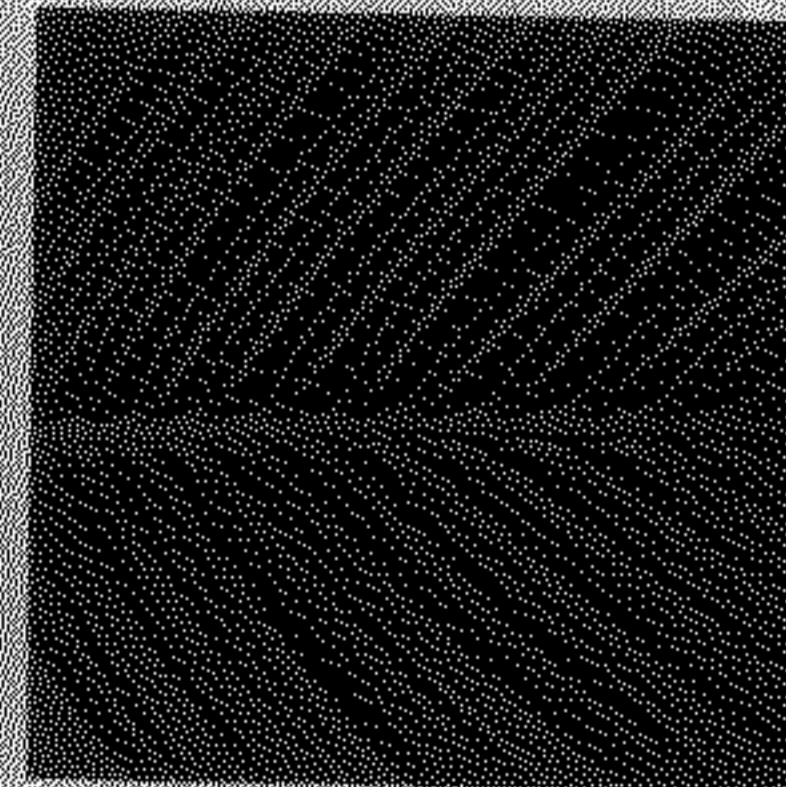
KID12.JPG



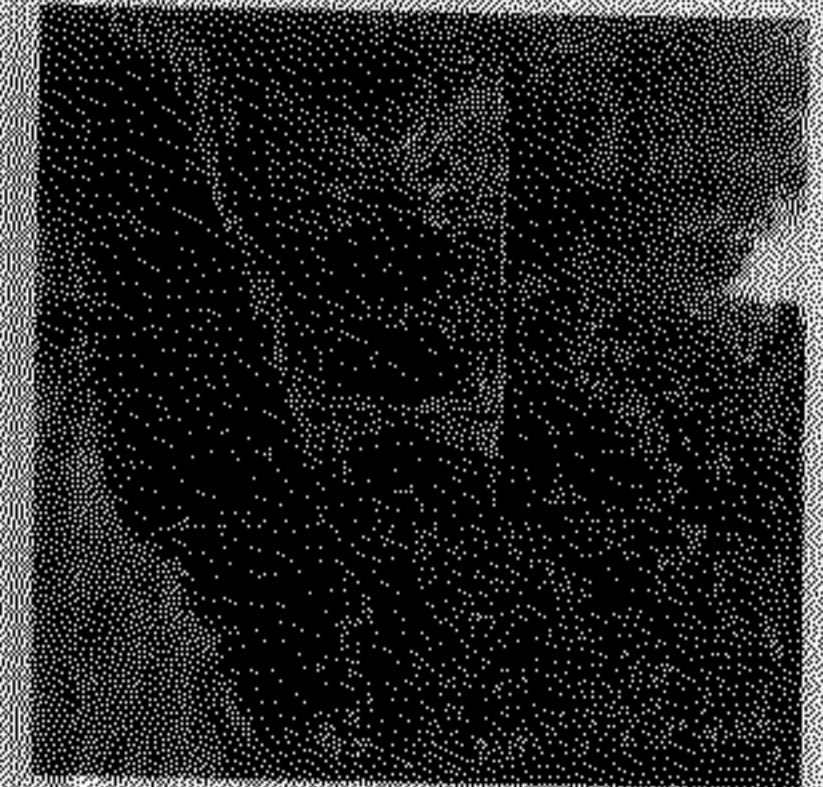
KID22.JPG



KID32.JPG



LEAF2.JPG



NEWENG2.JPG

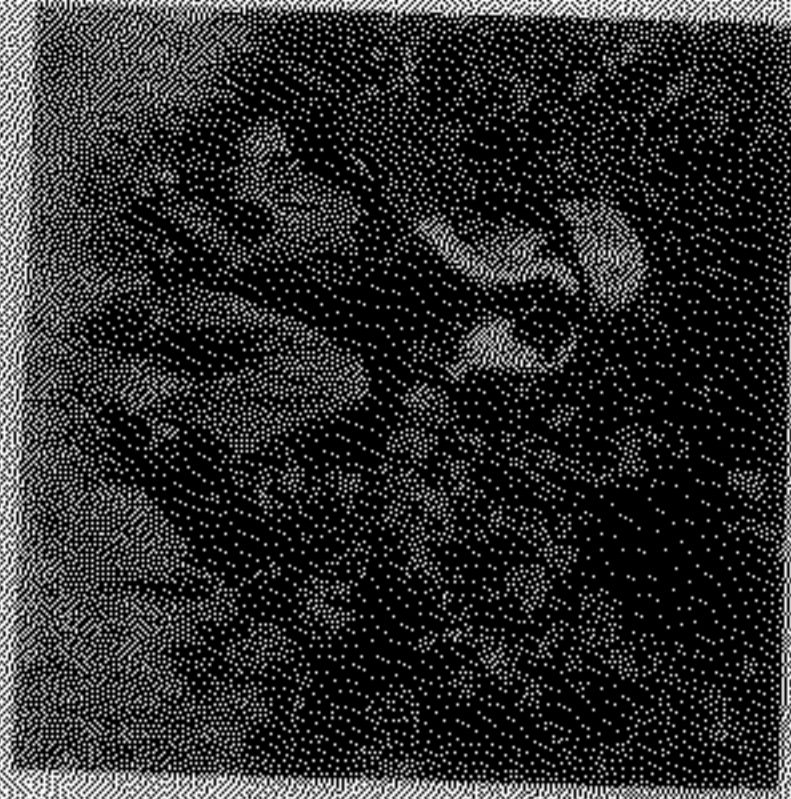
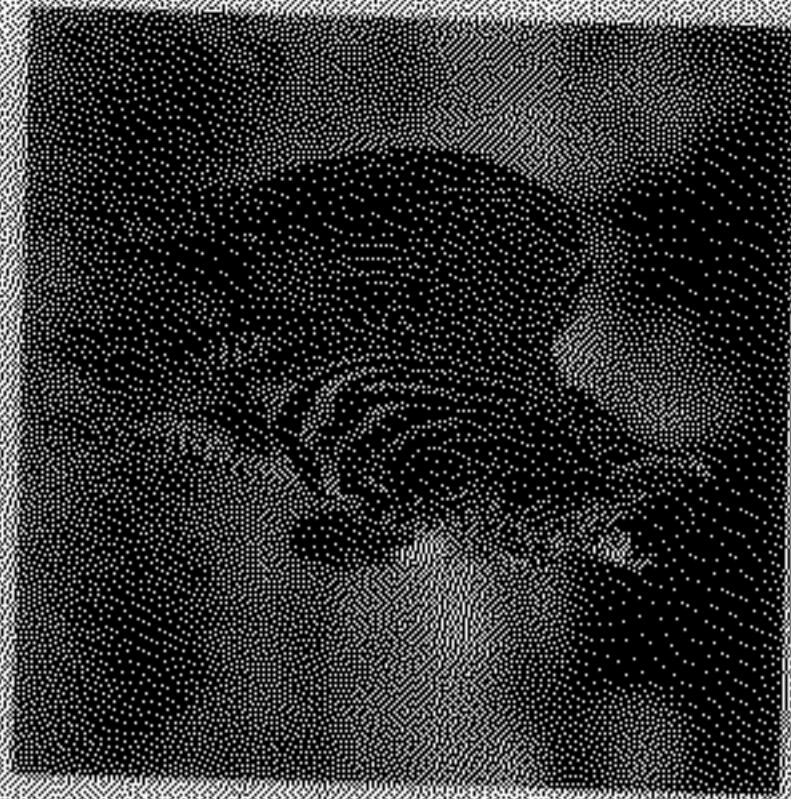
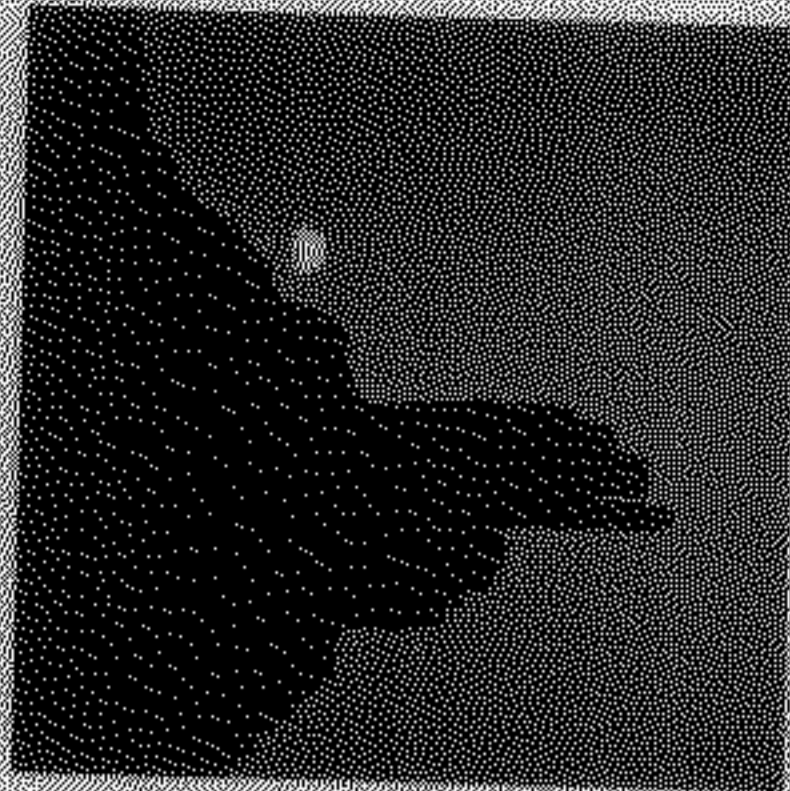


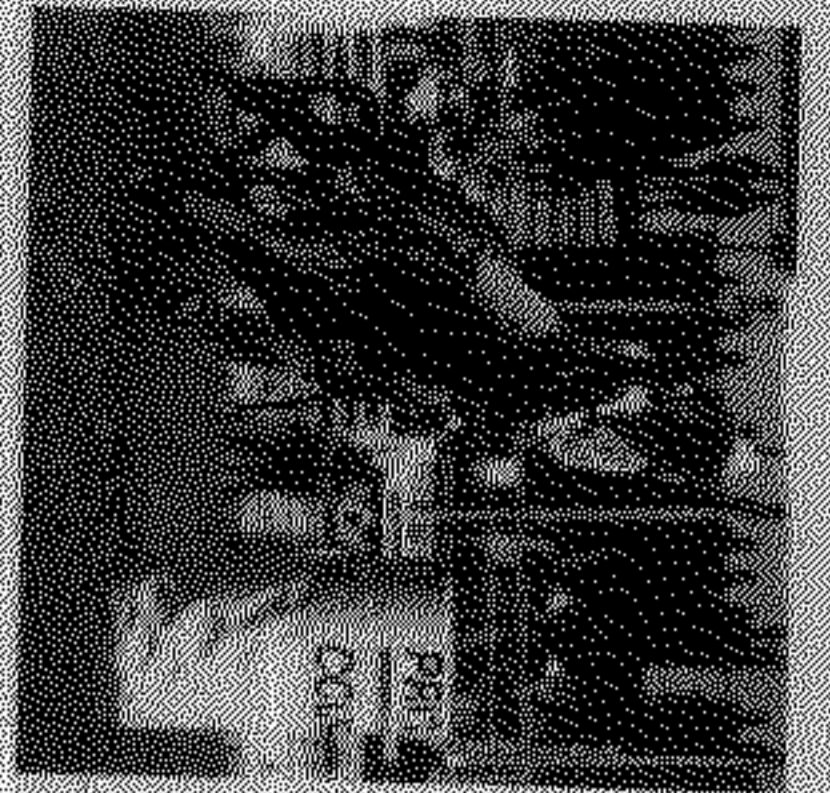
PHOTO2.JPG



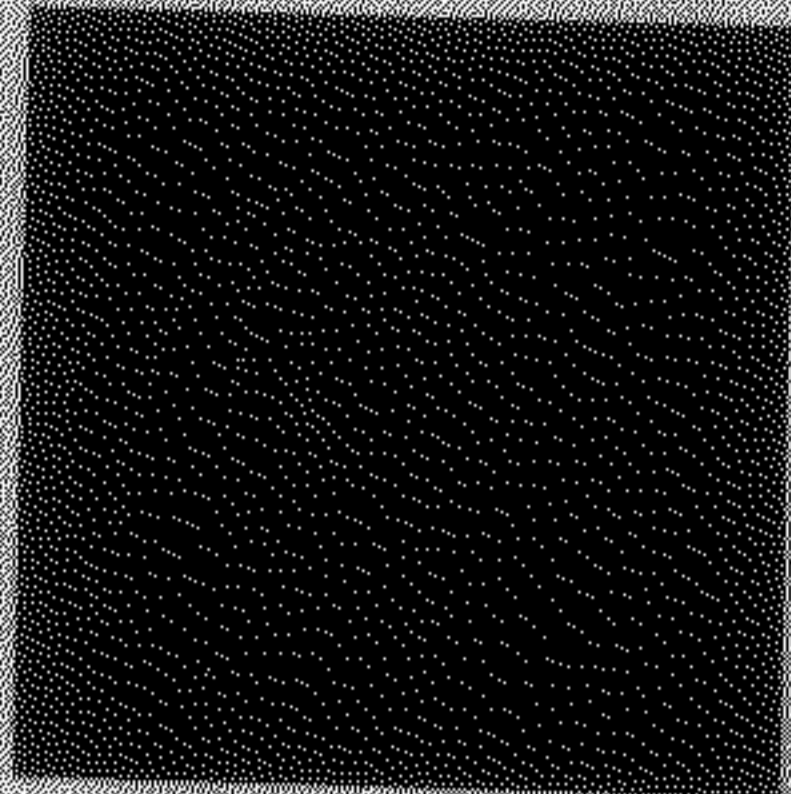
RGB32.JPG



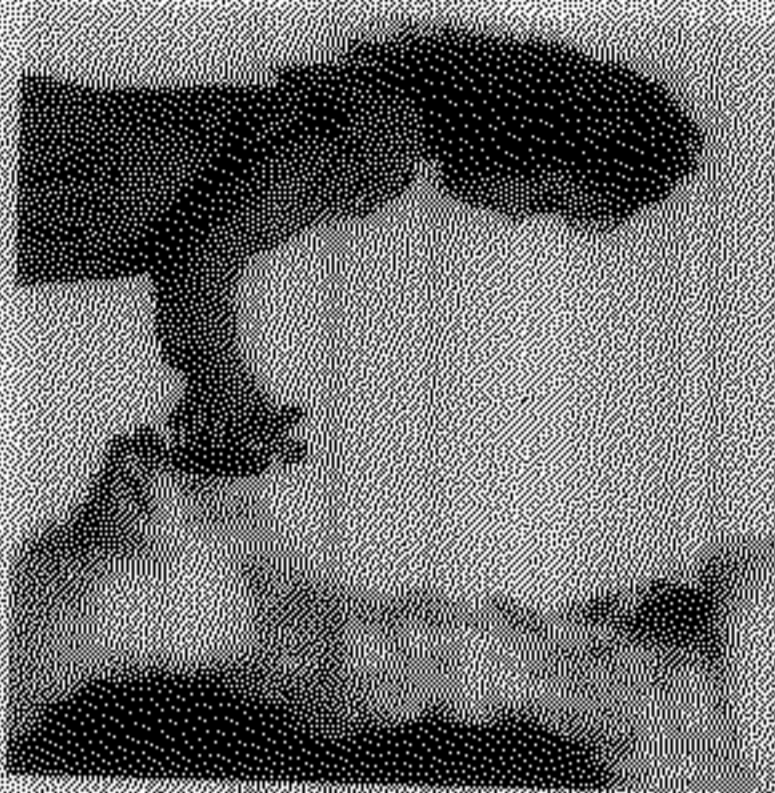
ROCK2.JPG



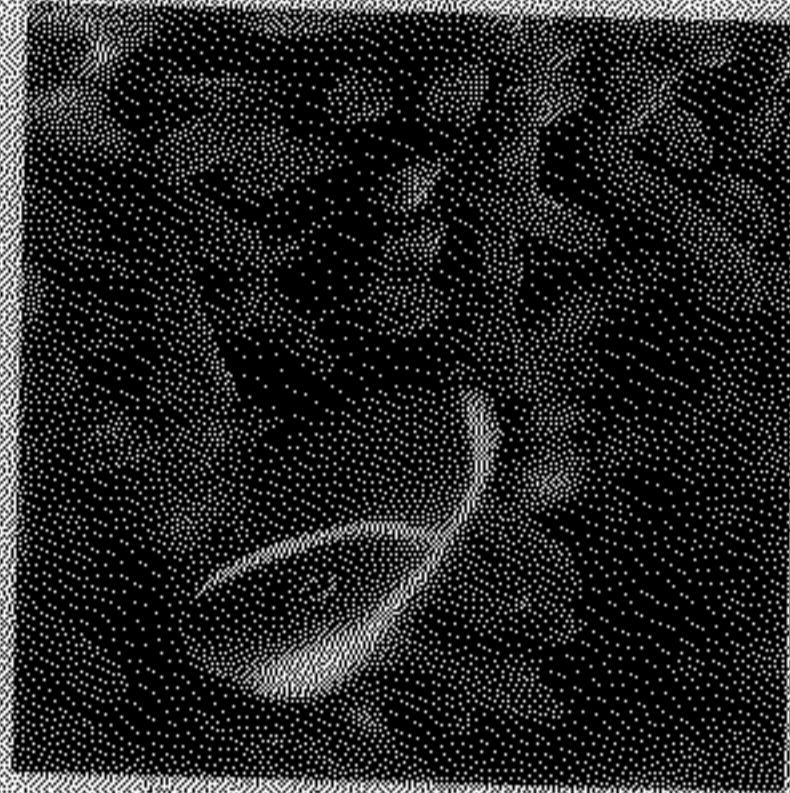
RODEO2.JPG



ROSE2.JPG



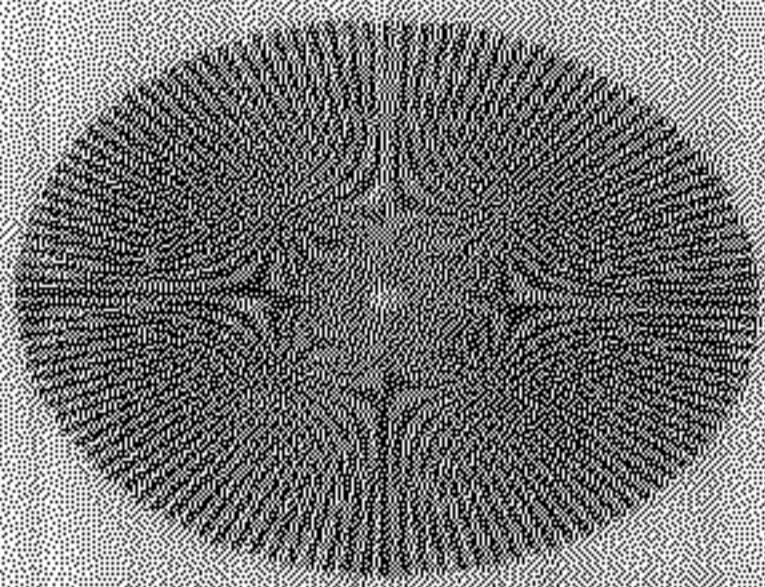
SANTA2.JPG



SEAFISH2.JPG



SOLDER2.JPG



STAR2.JPG

Chapter 4

Distance weighted Variance of grey-value (DWGV)

4.1 Introduction

Our main objective is to recognize images. Sometimes we may want to classify images according to their nature i.e. whether the subject of the images are same or not. Two images of bushes look similar, but one image of bushes and other of animal do not look similar. When two images are of same nature, their grey-level distribution is also of same nature. We made few experiments on this feature of images and got good results.

4.2 Preliminaries

When two images look similar, their grey-level distribution are also similar. We can find pixels of an image which has maximum grey-level values. We then compute the center of the image (x_c, y_c) using the formulae

$$x_c = \sum_i x_{mi} / n ; y_c = \sum_i y_{mi} / n;$$

where (x_{mi}, y_{mi}) , $i = 1(1)n$ are the co-ordinates of maximum grey-level valued pixels and n is the number of such points. Now **distance weighted grey-value variance (DWGV)** of all pixels is calculated w.r.t. the **maximum grey value**. The weight to each pixel is given by their distance from image center. This gives a feature for images.

Formula for DWGV :

$$DWGV = \frac{10000}{n \times (\text{row} \times \text{col}) \times D} \sum_{i=0}^{\text{row}-1} \sum_{j=0}^{\text{col}-1} (G_{ij} - G_m)^2 [(x_i - x_c)^2 + (y_j - y_c)^2]^{1/2}$$

where G_{ij} = grey value of (i , j)th pixel.

G_m = maximum grey value of the image.

D = distance of point at maximum distance from image center

$$= [(x_d - x_c)^2 + (y_d - y_c)^2]$$

(x_d , y_d) is at maximum distance from image center.

Theorem : DWGV is invariant under rotation/translation.

Proof : By rotation or translation the position as well as the grey value of any pixel remains unchanged w.r.t. the image, so DWGV will remain unchanged after rotation/translation.

4.3 Computing distance weighted variance of grey value

At first we compute the maximum grey value of the input image. We then note the co-ordinate of pixels having the maximum grey value. Now we find the center of the image, and calculate the distance weighted variance of the image about the center according to the formula given above.

Algorithm 4.3.1 : *Computing variance of distance of highest grey-level points*

Procedure VarMaxGrey(S, row, col)

/* S is the input image matrix,
row is the number of rows or vertical length of the image,
col is the number of columns or horizontal length of the image,
*/

begin

maxgrey = Maximum_Grey_Value_of_Image(S);

k := 0;

for i := 0 to row-1 do

for j := 0 to col-1 do

if(S[i][j] = maxgrey) then

begin

X[k] := i; Y[k] = j;

k := k + 1;

```

        end
    endfor
endfor

xc := 0; yc := 0;

for i := 0 to k-1 do
    xc := xc + X[i];
    yc := yc + Y[j];
endfor

xc := xc/k; yc := yc/k;

DWVAR = ComputeDistWgtdGrVal(row,col); /* calculate according to
the formula given above */

return;
end.

```

4.4 Experimental Results

```

/*****
OUT FILE OBTAINED FROM "file2.c" & "proj2.c" and over the
TAGRA Images (*.TGA).
*****/
Size of image (Row,Column) : 480 640

```

FILENAME	MAX GREY VAL	CENTROID	VALUE
armyl.tga	255	(309 , 320)	9.823565
blazel.tga	255	(263 , 385)	13.027952
castle1.tga	255	(451 , 349)	2670.471004
cathed1.tga	255	(387 , 365)	137.222901
chimpl.tga	255	(140 , 360)	2.702023
choper1.tga	255	(198 , 330)	481.788713
fish1.tga	255	(340 , 351)	40.714603
goldfish1.tga	255	(222 , 290)	16.426345
hawk1.tga	255	(201 , 249)	14.161354
ice1.tga	255	(90 , 283)	375.286889
insect1.tga	255	(368 , 345)	11.364667
kid11.tga	255	(179 , 221)	3.944026
kid21.tga	255	(179 , 172)	2.392170
kid31.tga	255	(293 , 536)	9.044316
leaf1.tga	255	(479 , 639)	34617.493046
newengl.tga	255	(20 , 212)	23.829639
photo1.tga	255	(342 , 276)	1818.407731
rgb31.tga	255	(187 , 424)	47.365645
rock1.tga	255	(303 , 198)	93.006744

rodeo1.tga	255	(278 , 493)	5.064832
rose1.tga	255	(479 , 639)	66589.046907
santa1.tga	255	(479 , 639)	51627.769189
seafish1.tga	255	(270 , 472)	10.379608
solder1.tga	255	(144 , 327)	85.160010
star1.tga	255	(236 , 310)	4.914920
sunset1.tga	255	(479 , 639)	9328.099197

/****** (ROTATED IMAGES) *****/

OUT FILE OBTAINED FROM "file2.c" & "proj2.c" and over the
TAGRA Images (*.TGA).

/******

Size of image (Row,Column) : 640 480

FILENAME	MAX GREY VAL	CENTROID	VALUE
army2.tga	255	(320 , 168)	9.806927
blaze2.tga	255	(386 , 214)	12.976175
castle2.tga	255	(321 , 103)	2786.568130
cathed2.tga	255	(362 , 101)	139.419367
chimp2.tga	255	(359 , 338)	2.714331
choper2.tga	255	(328 , 284)	479.092319
fish2.tga	255	(352 , 137)	40.612099
goldfish2.tga	255	(290 , 255)	16.508964
hawk2.tga	255	(250 , 276)	14.220951
ice2.tga	255	(281 , 393)	367.378433
insect2.tga	255	(343 , 109)	11.408110
kid12.tga	255	(222 , 299)	3.953612
kid22.tga	255	(173 , 299)	2.400152
kid32.tga	255	(537 , 185)	9.035187
leaf2.tga	255	(319 , 478)	21047.066766
neweng2.tga	255	(213 , 457)	23.867451
photo2.tga	255	(269 , 159)	1760.169615
rgb32.tga	255	(425 , 291)	47.199385
rock2.tga	255	(199 , 177)	93.561879
rodeo2.tga	255	(494 , 199)	5.053361
rose2.tga	255	(319 , 478)	45568.781048
santa2.tga	255	(319 , 478)	33764.800592
seafish2.tga	255	(473 , 207)	10.348899
solder2.tga	255	(325 , 335)	85.217084
star2.tga	255	(311 , 241)	4.947305
sunset2.tga	255	(319 , 478)	7486.311274

Chapter 5

Grey-Level Diffusion Value

5.1 Introduction

Our main objective is to recognize images. Sometimes we may want to classify images according to their nature i.e. whether the subject of the images are same or not. Two images of bushes look similar, but one image of bushes and other of animal do not look similar. When two images are of same nature, their grey-level distribution is also of same nature. Here we analyzed the images in 3-Dimensions (see fig 5.1). An image has certain grey value at each point on the image plane. If we plot the grey values of an image in Z-direction corresponding to every point on the image which is lying on X-Y plane, this looks like hills or mountains standing on a plane. Now the idea is to let every point on the mountain emit a ray in the direction normal to the plane surrounding the point and let the rays hit a plane parallel to X-Y plane. Now we measure the **normalized positional variance** of the points on the new plane where the rays hit. We made few experiments on this feature of images and got good results.

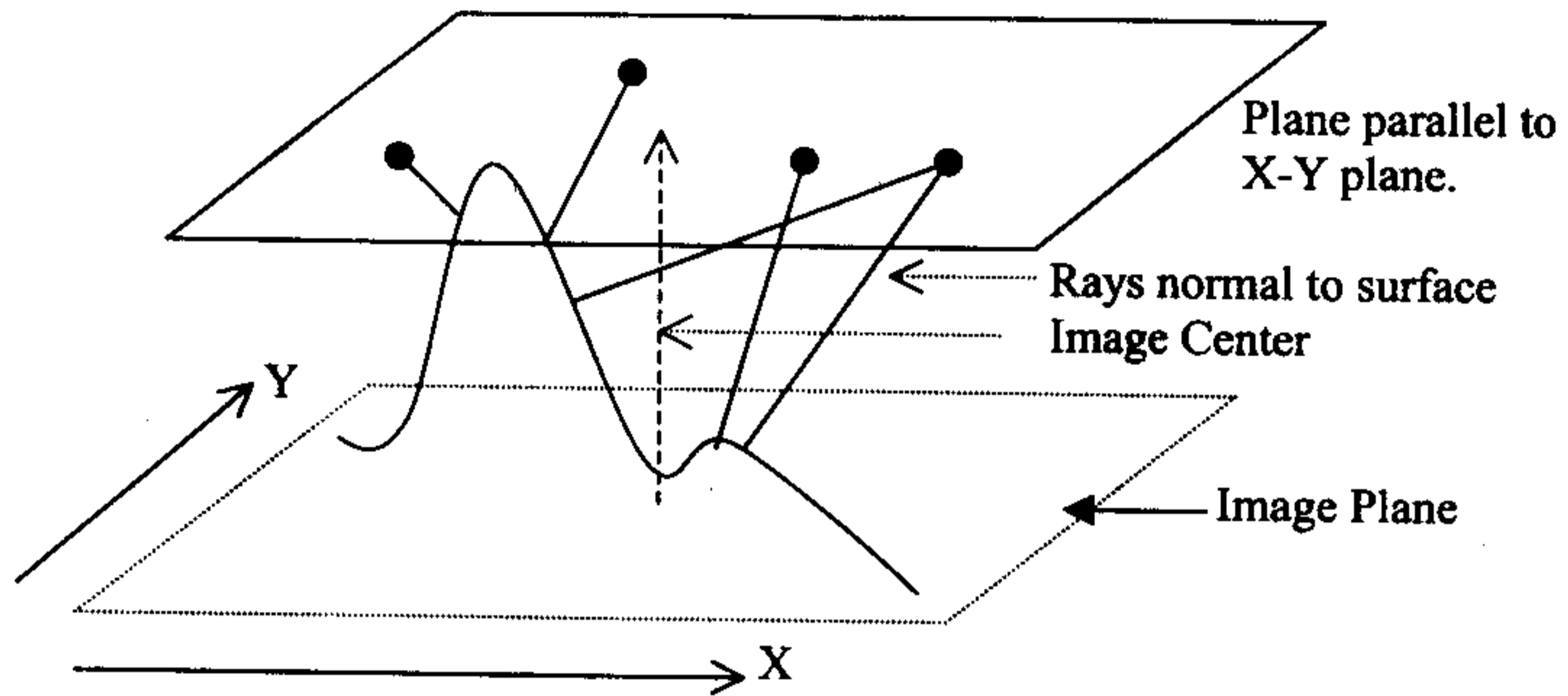


Fig . 5.1 : Illustration of Grey-Level Diffusion.

5.2 Definitions :

Normalized positional variance : It is defined as the variance of the distance about the image center divided by the maximum distance from the image center.

5.3 Preliminaries

Theorem : Grey-level diffusion value of an image is invariant under rotation , translation.

Proof : If we rotate and/or translate an image, the position of pixels remains unchanged about the image. So position of image center remains unchanged w.r.t. the image, thus the distance of the points on the projection plane from image center remains unchanged. Therefore grey-level diffusion value of an image remains invariant under rotation , translation.

5.3.1 Mathematical Tools

For our calculation we consider a right handed rectangular co-ordinate system.

Let \mathbf{i} : unit vector along positive X-direction.
 \mathbf{j} : unit vector along positive Y-direction.
 \mathbf{k} : unit vector along positive Z-direction.

Formulae :

- (i) $\mathbf{i} \times \mathbf{i} = \mathbf{j} \times \mathbf{j} = \mathbf{k} \times \mathbf{k} = \mathbf{0}$ (null vector)
- (ii) $\mathbf{i} \times \mathbf{j} = \mathbf{k}; \mathbf{j} \times \mathbf{k} = \mathbf{i}; \mathbf{k} \times \mathbf{i} = \mathbf{j};$
- (iii) $\mathbf{i} \times \mathbf{j} = -\mathbf{j} \times \mathbf{i};$
- (iv) If $\mathbf{A} = A_i \mathbf{i} + A_j \mathbf{j} + A_k \mathbf{k}; \mathbf{B} = B_i \mathbf{i} + B_j \mathbf{j} + B_k \mathbf{k};$

$$\text{Then } \mathbf{A} \times \mathbf{B} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ A_i & A_j & A_k \\ B_i & B_j & B_k \end{vmatrix} = (A_j B_k - A_k B_j)\mathbf{i} + (A_k B_i - A_i B_k)\mathbf{j} + (A_i B_j - A_j B_i)\mathbf{k}$$

- (v) If $A_i\mathbf{i} + A_j\mathbf{j} + A_k\mathbf{k} = B_i\mathbf{i} + B_j\mathbf{j} + B_k\mathbf{k}$ then
 $A_i = B_i ; A_j = B_j ; A_k = B_k;$

- If co-ordinate of 3 points are given we find the normal to the surface defined by these 3 points in the following way.

Let the points are $P_1(X_1, Y_1, Z_1), P_2(X_2, Y_2, Z_2), P_3(X_3, Y_3, Z_3)$.

$$\overrightarrow{P_1 P_2} = (X_2 - X_1)\mathbf{i} + (Y_2 - Y_1)\mathbf{j} + (Z_2 - Z_1)\mathbf{k}$$

$$\overrightarrow{P_1 P_3} = (X_3 - X_1)\mathbf{i} + (Y_3 - Y_1)\mathbf{j} + (Z_3 - Z_1)\mathbf{k}$$

$$\text{Normal : } \mathbf{n} = \overrightarrow{P_1 P_2} \times \overrightarrow{P_1 P_3} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ (X_2 - X_1) & (Y_2 - Y_1) & (Z_2 - Z_1) \\ (X_3 - X_1) & (Y_3 - Y_1) & (Z_3 - Z_1) \end{vmatrix}$$

- Formula : Position vector \mathbf{r} of any point on the line AB passing through the point A with position vector \mathbf{a} and parallel to the vector \mathbf{b} is given by
 $\mathbf{r} = \mathbf{a} + t\mathbf{b}$; where t is a scalar variable. (See fig 5.2)

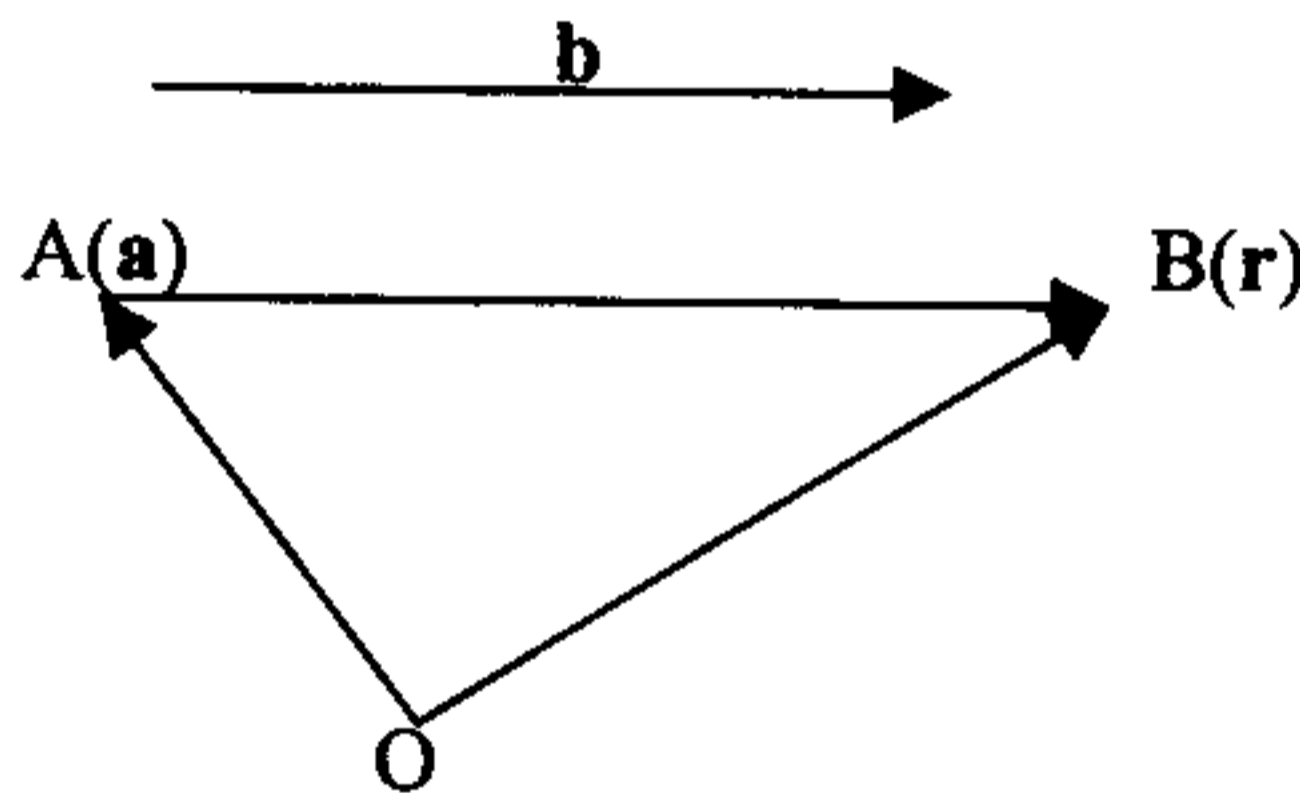


Fig 5.2

Say, we want to find normal at point P_1 . Let the normal will hit the projection plane at point $Q(X_q, Y_q, Z_q)$. Now

$$\begin{aligned} X_q &= X_1 + t[(Y_2 - Y_1)(Z_3 - Z_1) - (Y_3 - Y_1)(Z_2 - Z_1)] \\ Y_q &= Y_1 + t[(Z_2 - Z_1)(X_3 - X_1) - (Z_3 - Z_1)(X_2 - X_1)] \\ Z_q &= Z_1 + t[(X_2 - X_1)(Y_3 - Y_1) - (X_3 - X_1)(Y_2 - Y_1)] \end{aligned}$$

Now Z_q is the height of the projected plane, which is taken as the maximum grey value of the image. known constant and X_i, Y_i, Z_i $i = 1(1)3$ are known.

Thus, $t = (Z_q - Z_1) / [(X_2 - X_1)(Y_3 - Y_1) - (X_3 - X_1)(Y_2 - Y_1)]$

So X_q, Y_q, Z_q are found.

- Formula to find the center (X_c, Y_c) of the image

$$X_c = \frac{\sum_{i=0}^{\text{row}-1} X_i \sum_{j=0}^{\text{col}-1} G(X_i, Y_j)}{\sum_{i=0}^{\text{row}-1} \sum_{j=0}^{\text{col}-1} G(X_i, Y_j)}$$

$$Y_c = \frac{\sum_{j=0}^{\text{col}-1} Y_j \sum_{i=0}^{\text{row}-1} G(X_i, Y_j)}{\sum_{i=0}^{\text{row}-1} \sum_{j=0}^{\text{col}-1} G(X_i, Y_j)}$$

Diffusion Value :

$$DV = (1/N) \left[\sum_{i=0}^{\text{row}-1} \sum_{j=0}^{\text{col}-1} [(X_i - X_c)^2 + (Y_j - Y_c)^2] \right] / [(X_d - X_c)^2 + (Y_d - Y_c)^2]$$

where, $N =$ total number of points (row x col)

(X_d, Y_d) : co-ordinate of the point at maximum distance from the image center on the projection plane.

Maximum possible value of X_d and Y_d is $255 \times 255 = 65025$.

5.4 Computing grey-level diffusion value

Computing Normal to the surface concerning a image point : Any three points in space defines a plane. It is very difficult to predict which two points except the point of consideration are to be taken for calculation. If these points are chosen arbitrarily the measure may not be rotationally invariant. So we have used the following way to compute this (See fig 5.3).

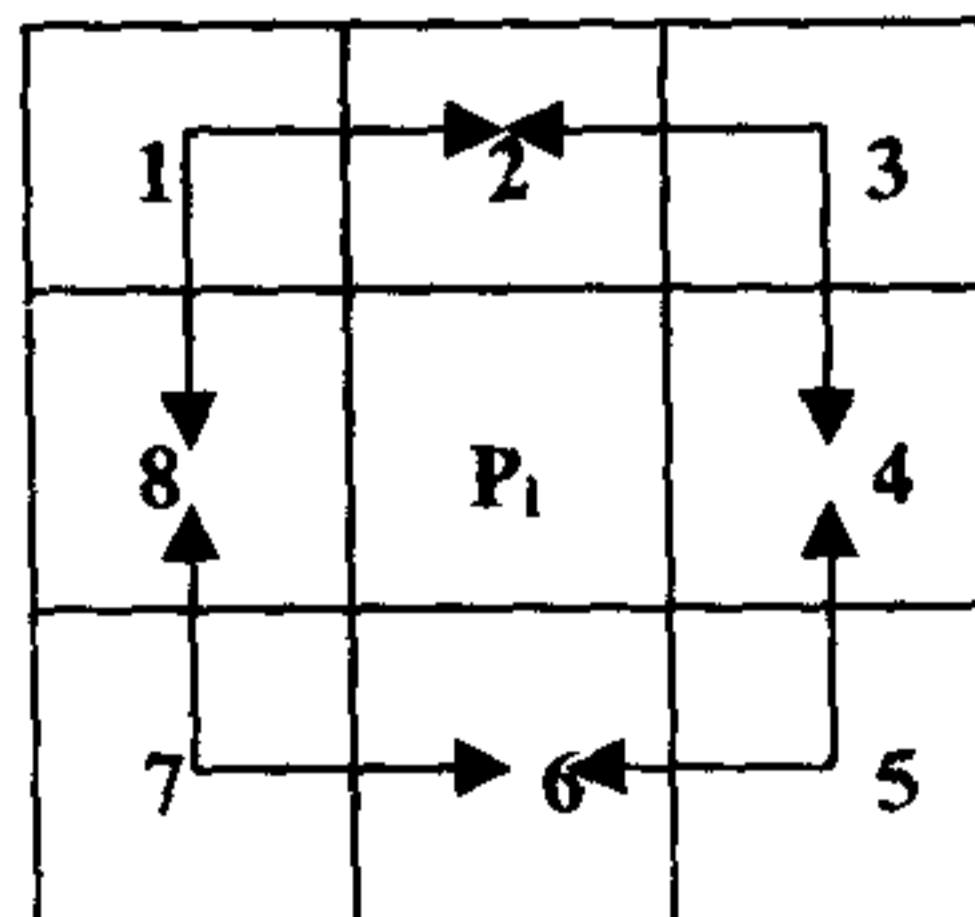


Fig 5.3

We calculate 4 normals taking 3-points at a time, they are (1,2,8); (3,2,4); (5,6,4); (7,6,8), and draw 4 lines parallel to these four normals passing through point P_i . We treat these 4 lines as 4 normals from point P_i and let them hit to the projection plane.

Algorithm 5.4.1 : Computing grey-level diffusion value

Procedure ComputeDiffValue(S, row, Col)

/* S is the input image matrix,
row is the number of rows or vertical length of the image,
col is the number of columns or horizontal length of the image,
*/

begin

 Compute image center (X_c , Y_c).

$Z_q = \text{ComputeMaxGreyValue}(S)$; /* height of projected plane */

 for i = 1 to row -2 do

 for j = 1 to col -2 do

 Compute and store locations ($Q_x[k]$, $Q_y[k]$) of 4 points on the projected plane which are hit by 4 normals from point $S[i][j]$, also note the co-ordinate of point(s) (X_d , Y_d) with maximum distance from (X_c , Y_c)

 endfor

 endfor

$DV = \text{CalculateDiffValue}()$; /* using the formula of DV */

 return;

end

5.5 Experimental results

```
*****/
OUT FILE OBTAINED FROM "file4.c" & "proj4.c" and over
the TAGRA Images (*.TGA).
*****/
Size of image (Row,Column) : 480 640
```

FILENAME	CENTROID	VALUE
armyl.tga	(266 , 314)	2.185231
blazel.tga	(218 , 346)	1.041019
castlel.tga	(229 , 339)	4.568130
cathedl.tga	(239 , 334)	4.717816
chimpl.tga	(188 , 355)	3.679135
choperl.tga	(193 , 338)	3.887817
fishl.tga	(259 , 343)	2.681433
goldfishl.tga	(267 , 250)	2.204353
hawk1.tga	(243 , 280)	1.397021
icel.tga	(184 , 324)	3.069224
insectl.tga	(240 , 340)	1.665033
kid1l.tga	(226 , 304)	2.244071
kid2l.tga	(220 , 301)	3.360692
kid3l.tga	(253 , 311)	1.750604
leafl.tga	(249 , 339)	2.549981
newengl.tga	(227 , 292)	5.830454
photol.tga	(289 , 309)	4.348589
rgb3l.tga	(249 , 332)	3.153349
rockl.tga	(186 , 303)	0.779668
rodeol.tga	(256 , 352)	2.658918
rosel.tga	(240 , 328)	0.281841
santal.tga	(223 , 319)	2.272003
seafishl.tga	(242 , 324)	1.228714
solderl.tga	(208 , 318)	4.172575
starl.tga	(239 , 322)	5.973378
sunsetl.tga	(233 , 313)	2.535313

/***** ROTATED IMAGES *****/
 OUT FILE OBTAINED FROM "file4.c" & "proj4.c" and over
 the TAGRA Images (*.TGA).

/*****

Size of image (Row,Column) : 640 480

FILENAME	CENTROID	VALUE
army2.tga	(314 , 213)	2.073447
blaze2.tga	(346 , 261)	1.145902
castle2.tga	(340 , 250)	4.602276
cathed2.tga	(334 , 240)	4.748223
chimp2.tga	(355 , 290)	3.638443
choper2.tga	(337 , 285)	3.961723
fish2.tga	(343 , 219)	1.738296
goldfish2.tga	(250 , 212)	3.356736
hawk2.tga	(280 , 236)	1.731179
ice2.tga	(324 , 294)	2.223029
insect2.tga	(339 , 238)	2.229364
kid12.tga	(304 , 253)	2.201501
kid22.tga	(301 , 259)	4.334070
kid32.tga	(311 , 226)	1.794460
leaf2.tga	(338 , 230)	1.637974
neweng2.tga	(292 , 253)	5.020532
photo2.tga	(309 , 191)	4.394399
rgb32.tga	(332 , 231)	2.722346
rock2.tga	(303 , 292)	1.413704
rodeo2.tga	(352 , 223)	3.062215
rose2.tga	(328 , 239)	0.277976
santa2.tga	(320 , 256)	2.322583
seafish2.tga	(324 , 237)	1.581731
solder2.tga	(318 , 271)	4.223330
star2.tga	(322 , 240)	6.003699
sunset2.tga	(313 , 246)	2.714187

Conclusion

This is well understood that the Image features we have proposed can serve the purpose of image recognition. We have already proved our measures are invariant under rotation and/or translation. The first measure , CEN & VEN, as we have proved, it is invariant under “rubber sheet” transformation and/or filtering unless the resulting image get changed. However, our second measure DWGV is invariant under rotation and/or translation but we need more research on the measure to make it full-proof. Our third measure Diffusion Value is also a good measure and it actually shows the direction of approach to the image recognition. It also need more research.

As a conclusion of this dissertation work I would like to say that Image feature is very peculiar. If we want get a match between all similar images, it is likely to return few dissimilar images also. If we strictly want not to get any dissimilar image in return, it is likely that we can not recognize all similar images.

Bibliography

- [1] Briggs F.A., Hwang K., and Fu K.S., in "Multicomputers and Image Processing – Algorithms and Programs" (Edited by K. Preston and L. Uhr), *Academic Press*, pp. 319-330, 1982.
- [2] Dagless E.L., Edwards M.D., Proudfoot J.T., "Shared memories in the CYBA-M multiprocessors", *IEE proceedings Part E 130*, pp 116-124, 1983.
- [3] Duff M.J.B., "Computing Structures for Image Processing", *Academic Press*, 1983.
- [4] Duff M.J.B., "CLIP4, a large scale integrated circuit array parallel processor", *Proc. 3rd Int. Joint Conf. on Pattern Recognition*, pp. 728-733, Colorado (CA), 1976.
- [5] Flanders P.M., Hunt D.J., Reddaway S.F., and Parkinson D., "Efficient high-speed computing with the Distributed Array Processor", in *High Speed Computer and Algorithm Organisation*, Kuck D.J., Lawrie D.H., and Sameh A.H. (eds.), *Academic Press*, New York, 1977.
- [6] Flynn M.J., "Very High Speed Computing Systems", *Proc. of IEEE 54*, pp. 1901 - 1909, 1966.
- [7] Gonzalez R.C. and Woods R.E., "Digital Image Processing," *Addison-Wesley*, Reading, Massachusetts, pp. 504-506, 1993.
- [8] Greanis E.C. et. al., "The Recognition of Handwritten Numerals by Contour Analysis," *IBM J. Res. Dev.*, Vol. 7, No. 1, pp.14-21, Jan. 1963.
- [9] Jain A.K. and Vailaya A., "Shape-based Retrieval : A Case Study with Trademark Databases," *Pattern Recognition*, Vol. 31, No. 9, pp. 1369-1390, 1998.
- [10] Kruse B., Gudmundson B., Antonsson D., "Multicomputers and Image Processing - Algorithms and Programs", *Academic Press*, pp. 31-45, 1982.
- [11] Machlup F. and Mansfield U., "The Study of Information : Interdisciplinary Messages", *John Willey & Sons.*, 1983.
- [12] Minsky M. and Papert S., "Perceptrons : An Introduction to Computational Geometry," *The MIT Press*, p. 86, 1969.

- [13] Pass S., "The GRID parallel computer system", *Image Processing System Architectures (Edited by Kittler J. and Duff M.J.B.)*, Chapter 2, John Wiley & Sons Inc., 1985.
- [14] Pratt W.K., "Digital Image Processing", *John Wiley & Sons.*, 1978.
- [15] Rosenfeld A., Kushner T., Wu A.Y., "Image Processing on ZMOB", *IEEE Transaction on Computers*, C-31, pp. 943-951, 1982.
- [16] Rosenfeld A. and Kak A.C., "Digital Picture Processing", *Academic Press*, Volumes 1 and 2, 1982.
- [17] Sternberg S.R., "Cytocomputer real-time pattern recognition", *Proc. 8th Automatic Imagery Pattern Recognition Symposium*, pp. 205, 1978.