UNIVERSITAT POLITÈCNICA DE CATALUNYA

MERIT European Master of Research on Information and Communication Technologies

Design and Implementation of Real-time Image Processing Algorithms for FP camera

by

Andrés Frías Velázquez

A dissertation submitted in partial fulfillment of the requirements for the degree of:

Master of Science

in

Research on Information and Communication Technologies

Advisor:
PhD. Josep Ramon Morros i Rubio

TECHNICAL UNIVERSITY OF CATALONIA

Barcelona, Spain

Summer 2008

Abstract

Design and Implementation of Real-time Image Processing Algorithms for FP camera

by

Andrés Frías Velázquez

Technical University of Catalonia

PhD. Josep Ramon Morros i Rubio, Advisor

A novel approach to perform gray-scale morphological operations based on bitwise-plane decomposition is preseneted in this work. The project aims to investigate novel implementation architectures for standard image processing algorithms embedded on a Focal Plane (FP) camera. Unlike to typical autonomous vision systems, FP camera allows performing highly-parallelized processing both in real time and at full resolution that result hard to implement with current techniques. Consequently, this technology breakthrough opens new fields of research on the design and implementation of basic vision tools considering the restrictions of this new architecture.

As mentioned, the project mainly focuses on developing gray-scale morphological operations based on the bitwise decomposition of the images. This sort of decomposition contrast significantly with the classical threshold decomposition approach, which most of the current techniques are based on. As a result, the approach presented in this work not only serves to implement morphological filters but also may provide novel and interesting theoretical insights on mathematical morphology.

Acknowledgements

Firstly, I would like to thank to God for his care and protection at any moment of this adventure far away from home. Also, I am very grateful of having this chance to live abroad and grow as a person from another point of view. I am greatly indebted to my family for supporting me in this endeavor. In particular, I would like to acknowledge my parents, Agustín and Avelina, who have always cared about my education and encouraged me to work hard to accomplish my goals. To my sisters and brother, Irene, Domi and Juan, I would like to thank them for their advices, patience and love.

For making more enjoyable my staying in Belgium and Spain, I would like to thank to all Merit students for their friendship and introduce me to their culture, traditions and language. I think I will never forget the trips, meetings, and BBQs we had together. Special thanks to my friends Xuefeng, Hanif, Rodolfo, Sebastian, Tao, Li Li, Yang, Adolfo, Salman, and Lu.

I would like to acknowledge to the European Commission and the Erasmus Mundus programme for supporting my Master studies during these two years.

Finally, I want to thank to my advisor, Dr. Ramon Morros, for his support, guidance and trust along the development of this thesis.

CONTENTS

List of Figures

List of Tables

Nomenclature

b	Number of bits.
$\mathscr{b}$	Number of branch of the tree.
f_ℓ	Bitwise bitplane.
h_ℓ	Threshold decomposition bitplane.
I_{er}	Gray-scale eroded image.
I_ℓ^t	Transformed bitwise bitplane.
ℓ	Bitplane level.
$MCD_\ell^{\mathscr{b}}$	Binary mask of the MCD condition.
$NCD_\ell^{\mathscr{b}}$	Binary mask of the NCD condition.
$R_\ell^{\mathscr{b}}$	Bitplane accumulation of the MCD condition.
$\mathcal{R}_\ell^{\mathscr{b}}$	Erosion of the MCD region.
se	Structuring element.
$S_\ell^{\mathscr{b}}$	Bitplane accumulation of the NCD condition.
$\mathcal{S}_\ell^{\mathscr{b}}$	Erosion of the NCD region.
$\cup$	Union set operator.
$\cap$	Intersection set operator.
$\ominus$	Morphological erosion operator.
$\oplus$	Morphological dilation operator.
$\circ$	Morphological opening operator.
$\bullet$	Morphological closing operator.

Chapter 1
State-of-the-Art

"For since the fabric of the universe is most perfect and the work of a most wise Creator, nothing at all takes place in the universe in which some rule of maximum or minimum does not appear."

-Leonhard Euler-

1.1 Introduction

After developing basic research and being traduced to an algorithm which can be understood by a computer, there is further research which deals with the optimal implementation of the algorithm in accordance with the most suitable VLSI (*Very-large-scale integration*) architecture. This step is critical in the success of a research proposal, since the speed of processing and the utilization area of the chip might play a key role in its performance, as in the case of image and video processing fields.

Given the higher dimensional nature of the images and video, handle such a large amount of data generated per image it is not an easy task. Normally, images are treated by computers as a 1-dimensional signal block, where the data is serialized in order to ease its computing and fit it to the most common architectures. On the other hand, alternative approaches consist on processing the image under a parallel structure, which accelerates the processing time at the cost of increasing the chip area, power consumption and price. As a result, current architectures attempt to tradeoff these variables in some way in order to adapt the best suitable solution for a given application.

In recent years, vision systems on chip have started to gain some attention from the scientific community, which have been called as second wave of solid-state imaging chips [1]. These sorts of vision chips are parallel computers that attempt to imitate the neural circuitry in the retina. Unlike a "dumb" video camera that merely records light intensity, the human eye integrates biological neural networks into each retinal cell, providing real-time information processing as photons arrive. As a result, only high-level information, such as edges and motion, are passed up to the brain. Conventional CCD and CMOS pixel sensors can be used to perform the same motion or edge detection, but at the cost of large amounts of digital computation after the images are acquired. By imitating the functionality of the retina, vision chips can eliminate most of that added computing equipment.

The technology of a vision chip is based on the Cellular Neural Network (CNN) model, which is a mathematical model that attempts to mimic a biological neural network allowing communication only between neighboring cells. This model seems to be a strong candidate to solve the paradigm of designing VLSI image processing systems, specifically for pixel-parallel vision processing. According to [2], a single *analog programmable array processor* can perform some image processing tasks in real-time with less power budget and manufacturing cost than an array of several digital processors that are believed necessary to carry out such tasks with floating point accuracy. Consequently, analog processors seem to be a complementary solution for vision systems given that the image can be processed in parallel at full resolution as the image is acquired in the sensor, similarly as the retina does.

The architecture of the analog array processor is not rigid, in the sense that strictly speaking is a mixed signal processor, since it combines non-linear analog processing with logic operations. This flexibility allows us to take advantage from both analog and digital world. Several applications have been envisioned to be suitable under this architecture, which includes security systems, autonomous robots, artificial implantable retinas and biochemical analysis.

As discussed above, vision chip systems have very attractive features that enable them to process images at very high speed; however, it is also convenient to point out that this type of bio-inspired vision system suffers the same drawbacks that every analog system possesses. That is, the degradation of the signal along the time and the high sensitivity to noise. These two factors are the main reasons by which nowadays our life is predominantly a digital world. Then, coming back to the analog world might be considered as a lagging by some people; even though, analog and mixed signal techniques has been improved significantly in the past decades, which have made digital system designers reconsider about integrating analog circuitry in their systems.

Although the state-of-the-art of analog array processors devoted for vision systems is relatively new, it has been maturing rapidly in the past 5 years. Currently, two prototypes have been launched for research purposes and simple industrial applications; one of them is called SCAMP™ (SIMD Current-Mode Analogue Matrix Processor), which is developed by the University of Manchester. Some of the features that stand out in this camera are: 3x3 kernel convolution filtering, Sobel edge detection, smoothing, active contours, skeletonization, and basic block matching motion field estimation. On the other hand, Anafocus, a spin-off of the Institute of Microelectronics of Seville, has created a vision system called Eye-RIS^{TM}, which supports the same features of gray scale filtering, a wide range of binary morphological operations, active points and blob handling functions.

In both systems only binary morphological operators are supported at the moment. The relevance of preserving gray-scale information of an image and avoid thresholding at early processing stages is crucial for diverse applications. Therefore, expanding the basic morphological operators to gray-scale format seems to be a logical next step; nevertheless, this task is not trivial as one might expect. Most of the current approaches that compute gray-scale morphology exploit the binary decomposition of the image to apply morphological operators onto a row of data rather than a set of bitplanes. As a result, new techniques have to be formulated considering the restrictions imposed by the analog world and adapted to this new parallel architecture.

Even though, this technology has just surfaced in recent years, at the end, combining analog and digital technologies seems to be a good solution that provides the best tradeoff between reliability and high speed performance for vision applications.

1.1.1 Objectives

This work aims to investigate novel implementation architectures for standard image processing algorithms embedded on a analog programmable array processor, also called Focal Plane (FP) camera. As treated previously, unlike to typical autonomous vision systems, FP camera allows performing highly-parallelized processing both in real time and at full resolution that result hard to implement with current techniques. Thus, the project mainly focuses on developing gray-scale morphological operations under this new architecture.

1.1.2 Justification

Morphological transformations are commonly used to perform a variety of image processing tasks such as fillets, holes, wedges, and cracks by operating with various shape-structuring elements (*See* [3]). Their applications range at different levels of complexity, from industrial inspection to fast object recognition and image segmentation. However, morphological operators involve time-consuming processing given that they require data ordering and min/max computation resulting from the interaction between image and structuring element. Furthermore, implementation difficulties arise when an image is processed with a large-size gray-scale structuring element.

Morphological processing on binary images is simpler because set operators can be implemented by using logic AND/OR functions. Conversely, it is harder for gray-scale images because intersection and union of shapes are implemented as maximal or minimal value selection after addition or subtraction.

There exist several approaches in the literature that deal with the implementation of gray-scale morphology. The early attempts to implement morphological operations on hardware are the so-called pipeline architectures [4]-[6], these approaches intend to carry out binary and gray-scale morphology straightforward by performing additions and comparisons among the elements affected by the structuring element. These proposals are considerably inefficient given the excessive hardware employed to compute the addition and its increasing complexity when the structuring element becomes large.

An important contribution called threshold decomposition was introduced by Fitch et al. [7], which enabled performing gray-scale operations through the decomposition of any k-level signal into a set of k -1 binary signals. This sort of decomposition intends to serve as a superposition principle for non-linear transformations. In this way, this approach can be extended to images and being used to manipulate gray-scale images through binary transformations, which are usually faster and easier of being implemented with logic gates.

From a practical standpoint, performing image morphology in parallel (i.e. at a bitplane level) is a difficult task since the hardware area required tends to be very bulky given the 2D nature of images. Therefore, approaches based on serialization have been proposed in order to relax the problem. Serial design seems to be more flexible than pipeline design, but unfortunately with a longer cycle to get the final result.

In order to ease the computing and take advantage of the threshold decomposition, an hybrid "parallel-pipelined" approach was proposed by Shih et al.[8]. This algorithm actually aims to apply morphological operations on gray-scale data through a sliding window. This window contains the input data affected by the kernel, then it is applied the threshold decomposition, and each resulting binary level is processed in parallel by using simple logic and shift registers. At the end, the morphological transformation is obtained by adding the results obtained on every binary level. Other similar approaches are presented in [9] and [10].

Until now we have seen that no digital approach is able to perform *real parallel* morphological operations in a feasible way. Focal plane processors open a new line of research in which gray-scale morphological processing can be performed at a plane level rather than at serial or windowing level. In this sense, we can take advantage of the already existent binary morphological library of the vision chips to extend it to gray-scale morphology and accelerate its processing.

1.2 The Focal Plane Processor

To meet computational demands of computer vision algorithms, particularly if cost, size, and power dissipation of the system are important, it is often beneficial to perform some image processing directly on the focal plane, using a smart-sensor device. Some simple low-level image processing tasks can be implemented using dedicated analog circuits embedded within each pixel of the image sensor array.

By assigning a single processor to each pixel of an image the inherent parallelism of low-level image processing tasks can be fully exploited. Specific hardware solutions, however, lack the flexibility of a software-programmable computer, specifically on its ability to implement a variety of complicated algorithms using relatively simple hardware. The main difficulty on implementing a software-programmable pixel-preprocessor vision chip is the very limited area available for each processor in the array. An interesting alternative to digital processors is provided by "analog" processors, derived from the CNN (Cellular Neural Network) architecture supported with analog and digital memories.

A vision chip employs switched current "analog microprocessors" as processing nodes in a digital-like massively parallel computer architecture. Using analog processing elements allows real-time image processing with high efficiency in terms of silicon area and power dissipation.

The architecture of a simple focal plane processor is depicted in Fig. 1.1. The vision chip is composed of an array of Smart Image Sensors (SIS), in which every element not only serves for light capturing, but also works as an analog processing element. As mentioned previously, each of these processing elements contains an Arithmetic Logic Unit (ALU), registers, flags and I/O control, but adapted in an analog fashion. Each pixel-processor may execute instructions in a local or global manner. This means that the user is able to indicate to certain cells in the array to perform a given task, while the others remain awaiting for an instruction. Note that when a local instruction is ordered, it is performed at once along the entire array or in the cells of interest.

Given that local memories of the analog processor degrade with time and certain control has to exist on the chip, a digital processor and memories are attached to the design to make it reliable. Another important advantage is that we can exploit dual processing, in the sense that while the digital processor performs image post-processing, the focal plane processor may capture the next image to be processed and remove noise for example.

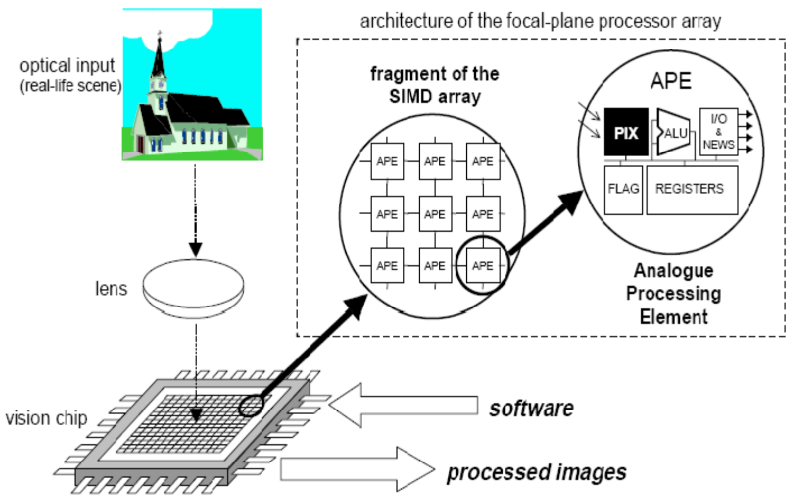


Fig. 1.1 Typical Focal plane processor.

The main objective of the focal plane processor is to carry out early image processing that is implemented more efficiently in the analog array such as image arithmetic, filtering and morphology. While, the digital processor is devoted to perform operations that by de facto result natural for this kind of processor such as recursions, shifts, as well as complicated time-frequency-spatial transformations, which may require high accuracy. The structure of this dual processing is depicted in Fig. 1.2. Note that after image post-processing the digital processor may appeal for feedback to the analog processor, which allows to the digital processor relax its workload at runtime. When this flow of information between processors occurs, we have to take into account that switching from digital to analog data representation and vice versa, introduces quantization errors that might become significant for certain applications.

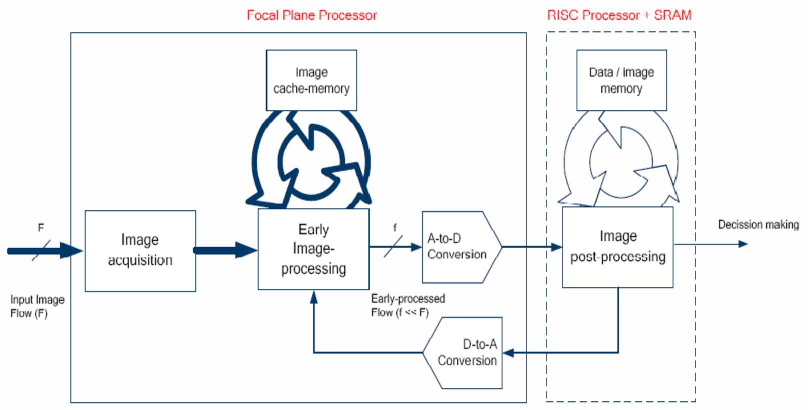


Fig. 1.2 Main components of a mixed signal vision system.

An important consideration that we have to keep in mind at every moment when working with analog systems, specifically with analog memories, is that these ones degrade along the time due to current leakage. Typically, they are conceived for retaining an image during several frame-times (the time between two consecutive frames) at 25fps frame rate. Figure 1.3 depicts the mean value of an image stored within an analog memory (expressed in LSBs) vs. storage time (expressed in milliseconds). As shown, the mean value of an image decreases around 1 LSB every 50ms. Furthermore, such degradation rate increases with the operation temperature.

The restriction imposed by the analog systems obliges to change the way in which the instructions of a digital processor are executed. This means that when the digital processor asks to the focal plane to perform certain computation, the programmer has to pass effective instructions that should be executed at a minimum time.

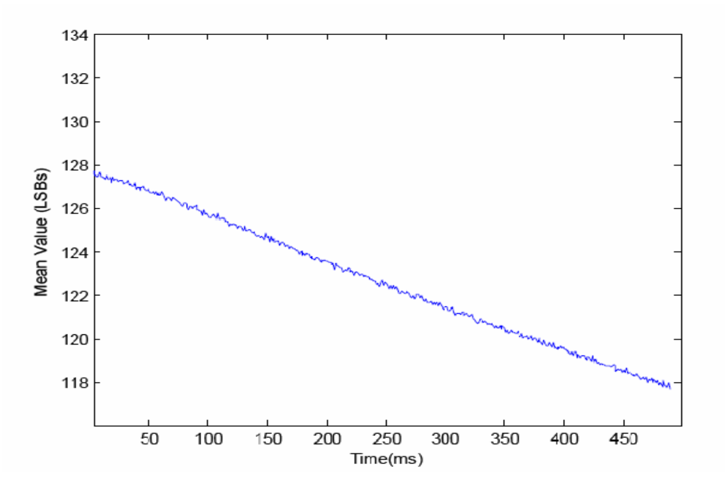


Fig. 1.3 Degradation of an analog memory along time.

1.2.1 Scamp Vision System

SCAMP (SIMD Current-Mode Analogue Matrix Processor) is a focal plane processor that is composed by a mesh connected array of processors, which are called APEs (analog processing elements). Several basic image processing algorithms have been implemented under this system, including convolution, linear and non-linear filtering, edge detection, basic segmentation, motion detection and estimation, histograming and binary mathematical morphology. Probably, the most attractive features of the SCAMP system are the ability of computing motion estimation and histograming. These two applications are quite challenging for any digital signal processor given the complexity of the algorithms involved and the time-consuming computing. According to [11], the motion estimator is based on a 21x21 global block search matching in the horizontal direction with a maximum displacement of ±3 pixels, which is performed in only in 46.4µs. On the other hand, the same author reports that his system is able to get a 64-bin histogram in just 205.6µs. These timing figures give us a good idea about the computing power of the analog processing for such challenging algorithms, since common digital computers run in several milliseconds.

Another important feature of the system is that the power dissipation of this chip is comparatively lower than the Eye-RIS vision system and several application-specific focal plane processors. Moreover, as the algorithmic program execution implies time-multiplexing of hardware resources, the APE area is not so much larger than the pixel area of many special-purpose vision chips, which implement algorithms in hardware.

One major concern for potential users of this "strange" system is that programming and controlling such a device might be a difficult task. However, there exists software that enables the control of the vision system, supports algorithm development, simulation, compilation and host communication. High-level instructions of the most common algorithms are already available for the user, which alleviates the painful work of programming two processors at low-level programming language.

1.2.2 Eye-RIS Vision System

The AnaFocus Eye-RIS v1.2 Vision System is a compact and modular vision system including all the elements needed for sensing images, enhancing the sensor operation, processing in real-time the image flow, interpreting the information contained in the image flow, and supporting decision-making based on the outcome of such interpretation [12].

The Q-Eye is a Quarter CIF (Common Intermediate Format) resolution fully-programmable smart image sensor. It consists of an array of 176 x 144 cells and a surrounding global circuitry. Each cell comprises multi-mode optical sensors, pixel memories, and linear and non-linear analog processors and binary processors. Also, the cells are interconnected each other in several ways with its 8 closest neighbors, allowing highly flexible, programmable, efficient, real-time image acquisition and spatial processing operations. Under Smart Image Sensor technology, each processor is merged with an optical sensor. This means that each pixel can both sense the corresponding spatial sample of the image and process this data in close interaction and cooperation with other pixels. In fact, Smart Image Sensors are also called focal-plane processors because they process images at the same physical layer where they are sensed.

The hardware components of the Eye-RIS system are divided basically in three layers.

1. The Anafocus Q-Eye Smart Image Sensor, which performs the image acquisition and the early image processing tasks through the focal plane processor.

2. The Altera NIOS II digital processor, which is in charge of controlling the operation of the Q-Eye as well as ports. Also, its duties concern with the post processing of the image and decision-making.
3. I/O ports. It includes a variety of digital input and output ports such as SPI, UART, PWM ports, GPIOs and USB 2.0.

An important advantage of the Eye-RIS vision system over SCAMP is precisely the ability of interfacing with different types of commercial and well-known industrial communication standards, which increase its versatility. In fact, according to the company, an Ethernet interface might be available in the system in order to control the system remotely. From the application developer's point of view, programming the Eye-RIS vision system provides the advantages that C/C++ programming language offer, since the programming of the digital and analog processor is based on these languages.

Table 1.1 presents the capabilities of both vision systems for comparison. In overall, the functionalities offered by both systems are very similar. The actual differences lie on the programmability of the parameters in a given function or algorithm. For instance, the Eye-RIS system allows setting the sigma value of a Gaussian smoothing as well as a range of values to define the mask of a convolution filter. On the other hand, SCAMP is able to compute difficult applications such as motion estimation and active contours, but fails to provide programming flexibility to the user in some of their applications and communication tools.

Table 1.1 Features of the SCAMP and Eye-RIS vision systems.

Feature	***SCAMP***	***Eye-RIS***
Resolution	128x128	176x144
Smooth using 3x3 convolution template	S	S
Sharpen using 3x3 convolution mask	S	S
Sobel Edge detection	S	S
Median Filter of 3x3 mask	S	S
Histogram with 64 bins	S	NS
Motion estimation	S	NS
Binary Morphology	S	S
Active Contours	S	NS
Blob handling functions	NS	S
Stand-Alone operation	NS	S
Ctrl and comm. interfaces	NS	S

Note: S - Supported, NS – Non-supported.

1.3 Gray-Scale Morphology

Mathematical morphology is a theory of image transformations and image functionals which is based on set-theoretical, geometrical, and topological concepts. [13]. There exist several motivations for using morphological filters in diverse applications related to image enhancement, detection, texture analysis, industrial inspection. On the one hand, these sorts of filters concede major importance to preserve, uncover, or detect the structure of image objects, which make them more suitable than linear filters for geometry based enhancement and detection. On the other hand, given their non-linear nature are highly efficient to suppress non-Gaussian noise.

Binary morphology was the basis of early morphology foundations and it is based on the geometrical relationship or *connectivity* of pixels that are deemed to be of the same class. The basic operators are erosion and dilation, which cause an image to increase or decrease in spatial extent, respectively. In association with logic operators such as AND, OR and NOT it is possible define other interesting operators like skeletonization, thinning, thickening and shrinking. Since the aim of this work is to introduce implementation techniques, it is advised to the interested reader knowing in depth about these topics to look at the surveys and tutorials presented in [3] and [14].

1.3.1 Basics on Mathematical Morphology

As previously stated, the two most common morphological operations are *dilation* ($\oplus$) and *erosion* (Θ). Specifically, in dilation, the center or active pixel is set to the maximum of its neighbors, and in erosion it is set to the minimum of its neighbors. Since these operations are often performed on binary images, dilation tends to expand edges, borders, or regions, while erosion tends to decrease or even eliminate small regions. On the other hand, if morphological operations are performed to gray-scale images, dilation will expand maximum values on the image (brighter areas), whereas erosion will expand darker areas. Obviously, the size and shape of the neighborhood used has a very strong influence on the effect produced by either operation.

The two processes can be done in tandem, over the same area. Since both erosion and dilation are nonlinear operations, they are not *invertible* transformations; that is, one followed by the other will not generally result in the original image. If erosion is followed by dilation, the operation is termed *opening* ($\circ$). If the image is binary, this combined operation will tend to remove small objects without changing the shape and size of larger objects. Basically, the initial erosion tends to reduce all objects, but some of the smaller objects will disappear altogether. The subsequent dilation will restore those objects that were not eliminated by erosion. If the order is reversed and dilation is performed first followed by erosion, the combined process is called *closing* (●). Closing connects objects that are close to each other, tends to fill up small holes, and smooths an object's outline by filling small gaps. As with the more fundamental operations of dilation and erosion, the size of objects removed by opening or filled by closing depends on the size and shape of the neighborhood that is selected.

In order to put in context the definitions presented above, we will revisit the mathematical expressions of such operations and their properties, which are defined as follows: Let the grayscale input signal be denoted by *f*, and the structuring element as *se* with size *N.* The dilation and erosion operations of *f* by *se*, respectively, are in general defined by equations (1) and (2).

$$(f \oplus se)[n] = \bigvee_{k=-\infty}^{\infty} (f[k] + se[n-k]) \tag{1}$$

$$(f \ominus se)[n] = \bigwedge_{k=-\infty}^{\infty} (f[k] - se[k-n]) \tag{2}$$

where $\vee$ denotes supremum (or maximum for finite *se*) and $\wedge$ denotes infimum (or minimum for finite *se*). Flat erosion (dilation) of a function *f* by a small convex set *se* reduces (increases) the peaks (valleys) and enlarges the minima (maxima) of the function.

Properties of Erosion and Dilation

Several morphological properties are usually applied in order to obtain more sophisticated operators or to decompose complex transformations. As a matter of fact, they serve as basis to manipulate symbolically the images, as the algebra serves for linear systems. For notation simplicity, spatial coordinates of a set are discarded.

Commutation:

$$f \oplus se = se \oplus f \tag{3}$$

$$f \Theta se \neq se \Theta f \tag{4}$$

Incresing, considering that $f \subseteq g$:

$$f \oplus se \subseteq g \oplus se \tag{5}$$

$$f \Theta se \subseteq g \Theta se \tag{6}$$

Duality:

$$\overline{f \ominus se} = \bar{f} \oplus \widetilde{se} \tag{7}$$

where $\widetilde{se}$ is the reflection of the structuring element.

Relationship with set operators:

$$(f \cap g) \oplus se \subseteq (f \oplus se) \cap (g \oplus se) \tag{8}$$

$$(f \cap g) \ominus se = (f \ominus se) \cap (g \ominus se) \tag{9}$$

$$(f \cup g) \oplus se = (f \oplus se) \cup (g \oplus se) \tag{10}$$

$$(f \cup g) \ominus se \supseteq (f \ominus se) \cup (g \ominus se) \tag{11}$$

Erosion and dilation by a pair intersection set:

$$f \oplus (se_1 \cap se_2) \subseteq (f \oplus se_1) \cap (f \oplus se_2) \tag{12}$$

$$f \ominus (se_1 \cap se_2) \supseteq (f \ominus se_1) \cup (f \ominus se_2) \tag{13}$$

Erosion and dilation by a pair union set:

$$f \ominus (se_1 \cup se_2) = (f \ominus se_1) \cup (f \ominus se_2) \tag{14}$$

$$f \oplus (se_1 \cup se_2) = (f \oplus se_1) \cup (f \oplus se_2) \tag{15}$$

Chain rules:

$$f \oplus (se_1 \oplus se_2) = (f \oplus se_1) \oplus se_2 \tag{16}$$

$$f \ominus (se_1 \oplus se_2) = (f \ominus se_1) \ominus se_2 \tag{17}$$

1.3.2 Threshold Decomposition (TD)

As mentioned previously, the transition between binary to gray-scale morphology is not trivial. In order to be able to give such a big step, it is necessary a tool to extend the binary properties to a multilevel decomposition. In traditional linear systems, any signal can be decomposed into simple sinusoids, which can be isolated, transformed independently, and then recomposed. Unfortunately, this property does not hold on non-linear systems and other alternative procedures have to be found to accomplish the separation of a non-linear signal. Fitch et al. [15] introduced a tool named *threshold decomposition,* which achieves decomposing a *k*-level signal into the sum of *k* - 1 binary signals. Each of them can be transformed through non-linear operators, and then recombined via another tool called (*Stacking*) to get the output of the filter as if the transformation had been taken place in gray-scale format.

Mathematically speaking, the decomposition is defined as follows: Let $a(n)$ be a multilevel discrete sequence of length L, in the range of $1 \leq n \leq L$, and for each n, $a(n)$ is quantized to one of the k *values 0, 1, …, k-1.* Then, the level i threshold decomposition of the original signal at point n is:

$$t_o^i(n) = \begin{cases} 1 & if\ a(n) \geq i \\ 0 & if\ a(n) < i \end{cases} \tag{18}$$

where $1 \leq i \leq k-1$.

This approach can be easily extended to images (2D), in the sense that each quantized level obtained by thresholding represents a binary image. The resultng set of binary images can be now manipulated independently with any morphological operator, that is:

$$x_s^i(n) = \mathcal{O}\{t_o^i(n)\}_{se} \tag{19}$$

The resulting gray-scale image is recomposed by using the *stacking* approach, as stated in (20).

$$y_s(n) = \sum_{i=1}^{k-1} x_s^i(n) \tag{20}$$

The method of threshold decomposition yields important theoretical insights in the manipulation of non-linear operators at gray-scale level. On the one side, it shows that non-linear transformations of multilevel signals are reduced to the much simpler analysis of binary signals. On the other side, the decomposition has an important impact from the implementation point of view, since the computation might be performed under a parallel or serial architecture.

1.3.3 Generalized Threshold Decomposition

An extension of the work of Fitch [15] that deals with the signal decomposition for non-linear transformations was proposed by Mitra [16]. The author presents a new approach that is more flexible than the simple threshold decomposition, which can lead to more efficient implementations of non-linear filters (rank filters) and provides some understanding about the properties of these kinds of transformations.

Some disadvantages on the VLSI implementation of the rank filters have been identified when binary threshold decomposition is applied. That is, the number of binary rank filters becomes very large for many practical applications and it doubles for each additional bit of the input signal.

The proposal of Mitra [16] called *generalized threshold decomposition* can lead to fewer sub-sequences compared to that obtained using conventional threshold decomposition and is capable to deal with bipolar signals. The approach states that for any *monotone increasing* decomposition function $f_i(\cdot)$ $1 \le i \le k-1$, the rank ordering of each sample of the input sequence is preserved in each of the decomposed sequences $u_i(n)$, which yields (21) fulfilling the stacking property.

$$\sum_{i=1}^{k-1} u_i(n) = a(n) \tag{21}$$

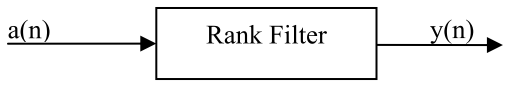
where $u_i(n) = f_i(a(n))$.

Fig. 1.4 shows how the input signal is splitted in several subsequences, not necessarily binary, and introduced to rank filters of lower order with a window of size *L*. This transformation may be expressed as follows:

$$y_i(n) = rank\_operator(u_i(n), \dots, u_i(n-L+1)) \tag{22}$$

The output of the filter can be recovered as stated in (23).

$$y(n) = \sum_{i=1}^{k-1} y_i(n) \tag{23}$$



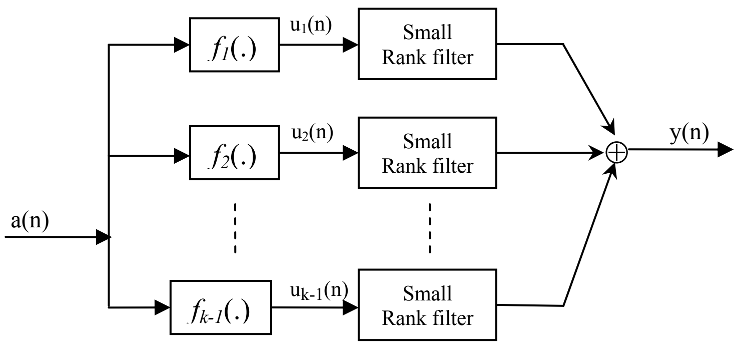


Fig. 1.4 Decomposition by the generalized thresholding property.

This generalized threshold decomposition allows splitting the multilevel input signal into several subsequences of lower dynamic range, not necessarily binary, and being processed separately with small rank filters. With this, the number of bits of the rank filter is reduced, which can be useful if the hardware being used performs better for smaller number of bits. The output is reconstructed by adding the results from all the filtered levels.

Observe that the binary threshold decomposition is a particular case of the more general decompositions for which the above theorem applies. Also note that the ***usual bitwise decomposition cannot be obtained through a monotone increasing function***; therefore, applying directly any morphological filter to this decomposition does not result in the correct morphological transformation.

A particular case that is treated in [16] is the Bit-related threshold decomposition, which is a mapping that fulfills threshold decomposition property and it is closely related to the bit-wise decomposition. For instance, assume that each sample of the input signal has *b* bits, which means a mapping between 0 and 2^b-1. Then, we can split the input signal with the following decomposition:

$$u_1(n) = \begin{cases} 1 & if\, u(n) \geq 2^{b-1} \\ 0 & if\, u(n) < 2^{b-1} \end{cases} \tag{24}$$

$$u_2(n) = \begin{cases} 2^{b-1} - 1 & if\, u(n) \geq 2^{b-1} \\ u(n) & if\, u(n) < 2^{b-1} \end{cases} \tag{25}$$

$$u_3(n) = \begin{cases} u(n) - 2^{b-1} & if\, u(n) \geq 2^{b-1} \\ 0 & if\, u(n) < 2^{b-1} \end{cases} \tag{26}$$

In this decomposition the input signal is divided in three parts. Equation (25) represents the sum of the binary decomposition from the levels in the range [1, 2^{b-1}-1), whereas (24) is just the binary level at 2^{b-1}. On the other hand, (26) is the sum of the binary decomposition from the levels in the range [2^{b-1}+1, 2^b-1]. This decomposition can be repeated iteratively to the sequences in (25) and (26), since they still remain as multilevel signals. Note that this decomposition reduces by two the dynamic range of the original multilevel signal, which is an interesting property that helps to implement non-linear transformations in a feasible way. Also, this property will be very helpful further to derive our approach.

It is worth to point out that there is a similar analysis that was developed independently by Zhang [17]. However, in this approach is asserted that there exists a mapping that allows non-linear transformation by using bit-wise decomposition, if certain conditions are fulfilled. Such constrain says that rank filters may be applied to a bit-wise decomposition only if the input signal belongs to the space: $\mathbb{D} = \{0, 1, 3, 7, 15, \dots\}$. Evidently, this restriction avoids using rank filters for diverse applications, since the space of numbers that is normally used is much larger, and hardly fit to this constrain.

Chapter 2

Morphology based on Bitwise Processing

"*If people do not believe that mathematics is simple,*
it is only because they do not realize how complicated life is "

-John von Neumann-

2.1 Bitwise Decomposition

As revisited in the last chapter, the *Threshold Decomposition* method allows implementing multilevel morphological filtering by using binary operators. However, three important inconveniences arise by using this approach if we desire to keep the complexity of the filters at a binary level:

1. The decomposition of the multilevel signal to binary is not performed in a typical bitwise representation. That is, thresholding should be applied to each level, which is an overhead in the processing.

2. As a result of the decomposition, the number of binary levels depends on the dynamic range of the input signal, which implies the same number of morphological filters. Thus, the complexity of processing in parallel becomes an issue when the dynamic range is large, especially if we are dealing with images.

3. The recomposition of the filtered signal requires performing numeric addition of all the binary sequences, which is a strong overhead since the complexity of the full-adder might be comparable to the morphological filter itself.

The usual bitwise decomposition of a multilevel signal into binary is an alternative approach that should be considered in order to minimize the overhead and make morphological filtering more efficient. This sort of decomposition offers attractive advantages over TD, since the overhead of decomposition/recomposition of the input/filtered signal can be drastically reduced by employing simple bit shifts and logical operators. Normally, multilevel data is stored in a bitwise fashion; therefore, the decomposition is performed in natural way since the complexity lies on reading the bit-fields or using bit shifts to extract the bits one by one. Of course, this depends on the flexibility of the processor's architecture. On the other hand, given that the bit fields are independent each other, the recomposition is accomplished by using both OR logical operator and bit shifts. By doing this we avoid using a full-adder, which is a burden for the threshold decomposition approach.

An example of the threshold and bitwise decomposition is presented in Table 2.1. We can note that by using bitwise decomposition the number of binary levels is only 3, whereas threshold decomposition requires 7. In this sense, threshold decomposition is a linear representation of the multilevel data; that is, the number of binary levels depends linearly on the dynamic range of the data. On the other hand, the bitwise decomposition is a logarithmic base 2 representation, which reduces significantly the number of binary levels.

Table 2.1 Example of TD and bitwise decomposition.

Bitwise decomposition						**Thresh. Decomposition**					
Data	3	7	4	2	0	Data	3	7	4	2	0
MSB	0	1	1	0	0	Top	0	1	0	0	0
	1	1	0	1	0		0	1	0	0	0
LSB	1	1	0	0	0		0	1	0	0	0
							0	1	1	0	0
							1	1	1	0	0
							1	1	1	1	0
						Down	1	1	1	1	0

Bitwise decomposition can be easily extended to images as shown in Fig. 2.1. These bit planes represent the decomposition of an 8 bit gray-scale image. Note that if we had performed the decomposition with TD, we would have obtained 256 bit planes. Moreover, if one bit is added to represent the gray-scale image, the number of bit planes is doubled.

Another important difference between threshold and bitwise decomposition is that bit planes obtained through the bitwise approach are hierarchically disposed according to their significance, whereas in TD every bit plane is equally significant. In overall, from a practical point of view, bitwise decomposition seems to be a better way to manipulate images at a binary level.

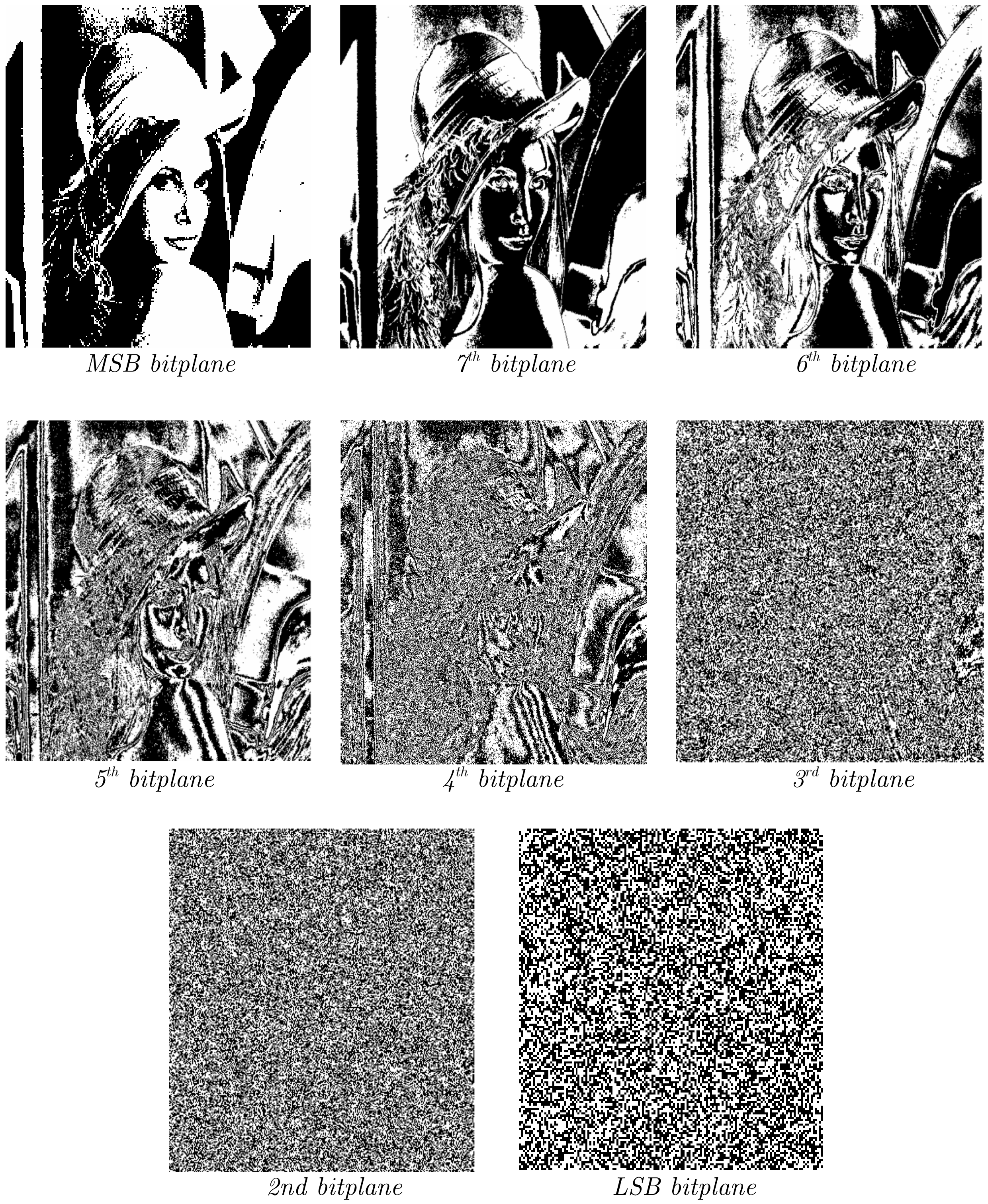


Fig. 2.1 Bitwise decomposition of a gray-scale image.

2.2 Bitwise Transformation

Bitwise decomposition/recomposition provides significant advantages over TD in terms of processing speed and minimum hardware overhead. However, as mentioned in the previous chapter, bitwise decomposition is not monotone increasing, which implies that does not follow the threshold decomposition method. That is, we cannot apply directly a morphological operator to each binary level and obtain the correct multilevel transformation by recomposing the processed binary data. Then, we have to look for alternatives approaches that allow processing the binary data under this scheme.

2.2.1 Binary Search Minimization

Binary search method is a well-known ordering technique in computer science and employed in diverse applications to rank finite sequences of integer data represented in a bitwise format. In order to illustrate its convenience and fast convergence let consider the following finite sequence:

$$A = \{14, 8,5,1,9,0, 11, 4,10\} \tag{27}$$

Since morphological filters are mainly based on max and min operators, we will focus on obtaining the minimum (erosion) of this sequence following the binary search method. First, note that the sequence A can be represented in a 4 bit format; then, at most 4 levels of decision will be required to reach the minimum. A level of decision is nothing more than the regions in which a sequence is thresholded given their bit of significance, that is $d = \{1, 2, 4, 8, ...\}$. The search is performed in a top-down fashion; thus, in the first level of decision we look for the elements that fulfill the condition stated in (28).

$$A < 8 = D_1 = \{5, 1, 0,4\} \tag{28}$$

Consequently, in the second level of decision we will get

$$D_1 < 4 = D_2 = \{ 1, 0\} \tag{29}$$

The third level of decision:

$$D_2 < 2 = D_3 = \{ 1, 0\} \tag{30}$$

Finally, the fourth level should provide us the minimum of this sequence:

$$D_3 < 1 = D_4 = \{0\} \tag{31}$$

Note that the search space is reduced significantly from one decision level to other, and in some cases the minimum can be found with less decision levels. On the other hand, the maximum of the sequence can be found only by replacing the inequality ($<$) by ($\geq$).

One important advantage of this method is that it can be implemented easily at a binary level by using logic operators and binary morphology. In the set of tables presented below is shown how we can reach the same result as the one obtained in (31). The first decision level is presented in Table 2.2(a). In this table we can see the bitwise decomposition of the data sequence ordered from the most to less significant level. Also, it is indicated what sort of rank operator is being applied to each binary level, whereas in the bottom shows how the data sequence is transformed.

For the first decision level, we apply the condition stated in (28). As a result, the boldfaced elements are the ones that fulfill such condition. Also, given that now our data is composed only by 0 and 1, we can replace the conditional inequality by a minimum binary operator. The minimum operation at this binary level yields a 0 in the erosion field.

Table 2.2(a) First decision level of the binary search minimization.

		Data sequence									
Level	Operator	14	8	5	1	9	0	11	4	10	Erosion
MSB- 1	Min	1	1	**0**	**0**	1	**0**	1	**0**	1	0
2	Min	1	0	1	0	0	0	1	1	0	-
3	Min	1	0	0	0	0	0	0	0	1	-
LSB- 4	Min	0	0	1	1	1	0	1	0	0	-
Transformed data		14	8	5	1	9	0	11	4	10	

Now, we proceed to apply the condition of the second decision level as we did in (29). Note that in such equation we discarded all the elements of the sequence except for {0, 1}. Then, we need some binary artifact to invalidate elements of sequence that are not candidates to be the minimum as we process each binary level. The logic OR operator allow us to do that when applied to the first and second levels of Table 2.2(a). The resulting transformation is substituted in the level 2 of Table 2.2(b) yielding the desired result. Now, the possible candidates to be the minimum is restricted to the elements highlighted in bold {0, 1}. On the other hand, the erosion at this level gives 0.

Table 2.2(b) Second decision level of the binary search minimization.

		Data sequence									
Level	Operator	14	8	5	1	9	0	11	4	10	Erosion
MSB -1	Min	1	1	0	0	1	0	1	0	1	0
2	Min	1	1	1	**0**	1	**0**	1	1	1	0
3	Min	1	0	0	0	0	0	0	0	1	-
LSB -4	Min	0	0	1	1	1	0	1	0	0	-
Transformed data		14	12	5	1	13	0	13	4	14	

For the third decision level, we apply the same procedure yielding Table 2.2(c). Note that the candidates to be the minimum are the same as in the previous decision level, which agrees with the results obtained in (30). Note also, that as we manipulate each binary level, the row *transformed data* also changes. The elements that have been discarded in previous levels increased their numeric value in the row transformed data, while the minimum candidates are kept untouched.

Table 2.2(c) Third decision level of the binary search minimization.

		Data sequence									
Level	Operator	14	8	5	1	9	0	11	4	10	Erosion
MSB- 1	Min	1	1	0	0	1	0	1	0	1	0
2	Min	1	1	1	0	1	0	1	1	1	0
3	Min	1	1	1	**0**	1	**0**	1	1	1	0
LSB- 4	Min	0	0	1	1	1	0	1	0	0	-
Transformed data		14	14	7	1	15	0	15	6	14	

Finally, in Table 2.2(d) the last decision level is presented. Note that the procedure of bitwise addition leads us to the same result presented above. In the erosion column of Table 2.2(d) the bitwise decomposition of the minimum value is finally obtained, (i.e. 0 for this specific case). After processing all the levels, the method is able to return the minimum of the sequence in two different ways. The former is called *straight,* which yields the bitwise decomposition of the minima (*e.g.* erosion column of Table 2.2 (d)), whereas the latter is called *referenced* because does not retrieve its value, but the location of the minima in the sequence (*e.g.* boldfaced element in the 4th level).

In this case, binary search minimization served as transformation to make possible applying binary erosion on each level. Therefore, in loose terms, we can state that the binary search minimization attempts to transform bitwise levels into a threshold decomposition equivalent. Note that this specific transformation is only valid if the non-linear filter to be applied is an erosion.

Table 2.2(d) Fourth decision level of the binary search minimization.

		Data sequence									
Level	Operator	14	8	5	1	9	0	11	4	10	Erosion
MSB- 1	Min	1	1	0	0	1	0	1	0	1	0
2	Min	1	1	1	0	1	0	1	1	1	0
3	Min	1	1	1	0	1	0	1	1	1	0
LSB- 4	Min	1	1	1	1	1	**0**	1	1	1	0
Transformed data		15	15	7	1	15	0	15	7	15	0

Binary Level Rectification

A special case of this method takes place when all the positions in a given binary level are set to 1, as shown in the third level of Table 2.3(a). Note that if we proceed with the bitwise addition till the fourth level, the final result is not correct. This is because in the third level there is no new information that might help us to discriminate between the potential candidates (i.e. {3, 2}) to be the minima. Therefore, the fourth binary level is transformed by applying the bitwise addition between the fourth level and the bitwise accumulation of the second level instead, yielding the desired result as shown in Table 2.3(b).

Table 2.3(a) Special case of no binary level discrimination.

		Data sequence									
Level	Operator	14	8	5	3	9	2	11	4	10	Erosion
MSB-1	Min	1	1	0	0	1	0	1	0	1	0
2	Min	1	1	1	**0**	1	**0**	1	1	1	0
3	Min	1	1	1	1	1	1	1	1	1	1
LSB-4	Min	0	0	1	1	1	0	1	0	0	-
Transformed data		14	14	7	3	15	2	15	6	14	

Table 2.3(b) Rectification of the 4th level of decision.

		Data sequence									
Level	Operator	14	8	5	3	9	2	11	4	10	Erosion
MSB-1	Min	1	1	0	0	1	0	1	0	1	0
2	Min	1	1	1	0	1	0	1	1	1	0
3	Min	1	1	1	1	1	1	1	1	1	1
LSB-4	Min	1	1	1	1	1	**0**	1	1	1	0
Transformed data		15	15	7	3	15	2	15	7	15	2

Binary Search Maximization

The maximization of a data sequence can be easily obtained by using the duality property stated in (7). In this sense, a few extra steps have to be done to get the maximum: 1) Apply the one's complement to every binary level. 2) Apply the method of binary search minimization. 3) Get the one's complement of the result obtained. The maximization can be also performed in a straightforward manner, avoiding the overhead of complementing the input and output data. However, bitwise addition (OR) and binary minimization (erosion) should be replaced by bitwise multiplication (AND) and binary maximization (dilation), respectively. This last statement is supported by the De Morgarn's theorem, which states (32) and (33).

$$A \cap B = \overline{\overline{A} \cup \overline{B}} \tag{32}$$

$$A \cup B = \overline{\overline{A} \cap \overline{B}} \tag{33}$$

Rank Searching

Binary search method not only can be used to find the maximum or minimum in a sequence, but also to extract a number according to its rank in the sequence. For instance, assume that our data sequence is composed by the set of numbers shown in (34). This sequence has been already put in order to better illustrate how the rank searching is obtained.

$$A = \{0, 1, 2, 3, 4, 5, 6, 7\} \tag{34}$$

Note that in the previous examples, the minimum operator was used in all the binary levels. However, max and min operators can be alternated to regard any element of the sequence. For convention, let regard 0 as the element ranked 1st (minimum), while 7 is the one ranked 8th (maximum) in the sequence. Suppose that we want to extract the element ranked 2nd. First, note that 3 binary levels are enough to represent the set of numbers in the sequence; then, the operators that we have to employ for the three binary levels is (min-min-max) as shown in Table 2.4(a). The order in which every operator is set provides the key to regard the element desired. In this case, the maximum operator in the least significant binary level allows extracting the 2nd ranked instead of the 1st ranked element.

Table 2.4 (b) shows how the ranked 2nd is obtained by using the duality property, since the 3rd binary level has been complemented and applied the process of binary search minimization. At the end, the result of minimizing of the third level is once again complemented, yielding now the correct ranked element.

Note that the elements ranked 3rd and 4th can be regarded by the following combinations (min-max-min) and (min-max-max), respectively. As a result, there exist 2^3 combinations, which each of them regard one element of the sequence *if this is full in range.*

Table 2.4(a) Binary search of the element ranked 2nd.

		Data sequence								
Level	Operator	0	1	2	3	4	5	6	7	Rank 2nd
MSB-1	Min	0	0	0	0	1	1	1	1	-
2	Min	0	0	1	1	0	0	1	1	-
LSB-3	Max	0	1	0	1	0	1	0	1	-

Table 2.4(b) Ranked 2nd element obtained by using duality property.

		Data sequence									
Level	Operator	0	1	2	3	4	5	6	7	Rank 2nd	Rank 2nd (comp,)
MSB-1	Min	0	0	0	0	1	1	1	1	0	0
2	Min	0	0	1	1	1	1	1	1	0	0
LSB-3	Min	1	**0**	1	1	1	1	1	1	0	1

2.2.2 Bitwise Image Morphology

Bitwise processing can be easily extended to images by using bitplane decomposition as depicted in Fig. 2.1. Now, our goal is to derive from each bit plane its corresponding morphological transformation at the same level of decomposition. In order to do so, each level is processed with a block called Bitwise Morphology Transformation (BMT) as shown in Fig. 2.2. Note that most of the blocks depend on the results obtained in the decision levels of greater significance, similarly as what happened when processing a data sequence.

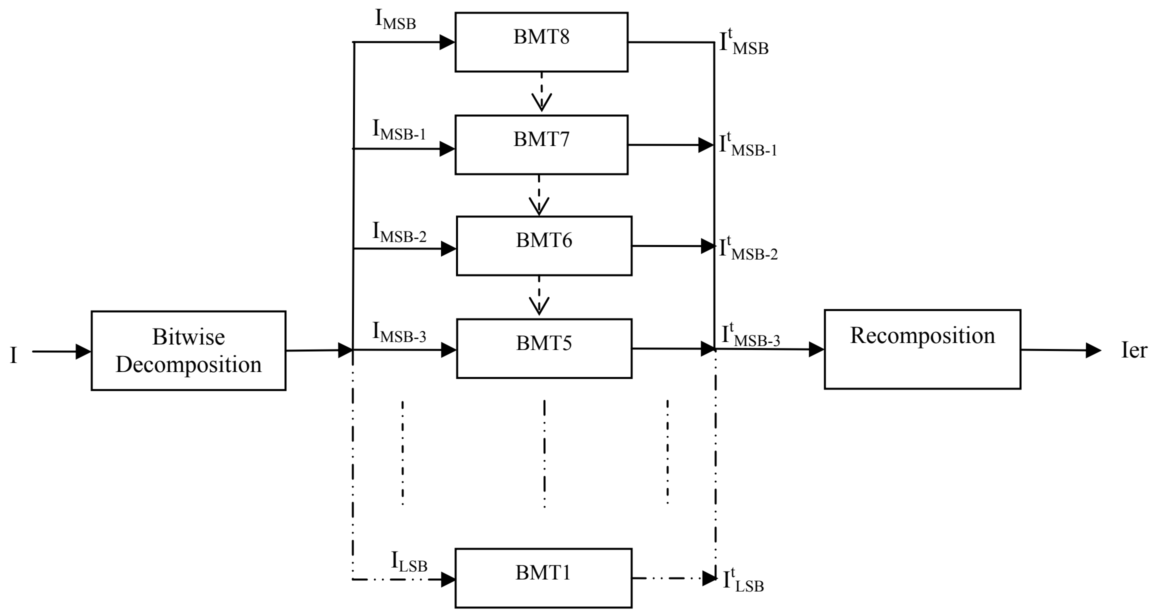


Fig. 2.2 Basic block diagram of a bitwise morphology processing.

Bitwise Morphology Transformation

The BMT block is the core of the algorithm, which relies on simple logic functions and binary erosion operator. This block intends to generate the bit planes of the eroded image by processing the corresponding bit planes in the original image.

Recall that, before using any morphological operator, certain image manipulation has to be done in order to accomplish the desired result. In the development of the algorithm we consider three basic assumptions that allow us to manipulate the bit planes:

1. The bitwise decomposition of the original image follows a hierarchical order in accordance with their bit of significance, which provides us information to discriminate the rank of each pixel.

$$I_{MSB} > I_{MSB-1} > \cdots > I_{LSB+1} > I_{LSB} \tag{35}$$

2. The bit plane transformation I^t at certain level ℓ (I^t_ℓ) only depends on the interested and upper significance bit planes.

$$I^t_\ell = \mathcal{F}\left(I_{\ell,} I_{\ell+1}, \dots, I_{MSB}\right) \tag{36}$$

3. As occurs in the binary search minimization for a data sequence, two possible cases might take place when we are manipulating a bitplane.

(a) *No Candidate Discrimination* (*NCD*): In this case, no information about the possible candidates to be the minimum is available along certain neighborhood. The set of neighborhoods that fulfill such condition compose a common area in the bitplane that we have called *NCD region.*

(b) *Minimum Candidate Discrimination (MCD):* In this case, there exists at least minimal information that let us discriminate in which positions of the neighborhood the minimum is likely located. As a result, this condition composes a complementary area to the NCD region, which we have named *MCD region.*

In practical terms, the *NCD* relates when all the positions in the neighborhood are set to 1, whereas the *MCD* when *at least* one of such positions is set to 0. Given that these two cases can take place at any bitplane, we have to consider them as the minimization search is performed, since binary level rectification has to be done only in the *NCD* case.

b-Level Tree Diagram

As we know, potential candidates to be the minimum are discriminated by applying the Boolean addition of consecutive levels. The result of such operation is accumulated and applied to the next level. When such bitwise accumulation does not provide information about the minimum, we appeal to the *binary level rectification*, which corrects the bitwise accumulation to the latest level where some information about the minimum was available. Therefore, this rectification oblige us to consider both NCD and MCD cases when each bit plane is processed, which lead us to decompose each bitplane in two or more subsets. This sort of processing produces a b-level tree structure as the one depicted in Fig. 2.3, where b denotes the necessary number of bits to represent the whole data. On the other hand, $S_{\ell}^{\mathscr{b}}$ and $R_{\ell}^{\mathscr{b}}$ represent the bitwise accumulation at ℓ-level and $\mathscr{b}$-branch, with or without binary level rectification, respectively.

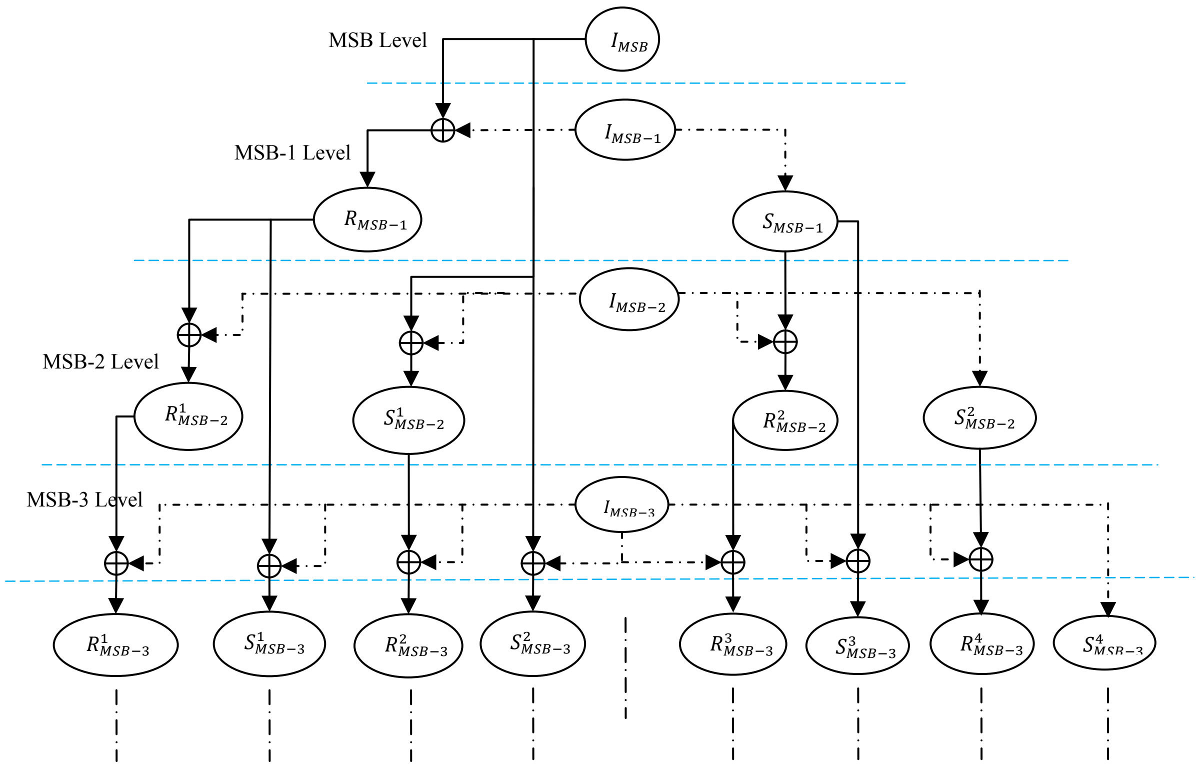


Fig. 2.3 b-Level tree diagram.

Note that this tree *only* describes the bitwise accumulation of the complete bitplanes according to the probable cases of discrimination for every branch and corresponding level, but not the actual NCD and MCD regions of every branch in which they take place. Such regions are, on the other hand, described in Fig. 2.4. At the left, the MSB-1 bit plane is divided into two regions. The gray shaded region NCD^1_{MSB-1} represents a binary mask that shows the set of values where the *NCD* case was fulfilled at the *MSB-1* level in the 1st branch. On the other hand, the mask MCD^1_{MSB-1} represents the *MCD* case. Consequently, each region is divided in two subregions in the next bit plane level, since the *NCD* and *MCD* cases might occur on each of the original regions. Finally, this sort of decomposition continues until the LSB plane level is reached.

Note that each bitplane is fragmented into independent regions, which agrees with the number of branches generated on each level of the tree presented above. Such segments will eventually serve as binary masks to extract the regions of interest of the accumulators (i.e. $S_\ell^{\mathscr{b}}$ and $R_\ell^{\mathscr{b}}$) after being processed.

Given the symmetry pattern of the algorithm, we can take advantage of such feature to construct iteratively the tree based on the example depicted in Fig. 2.4.

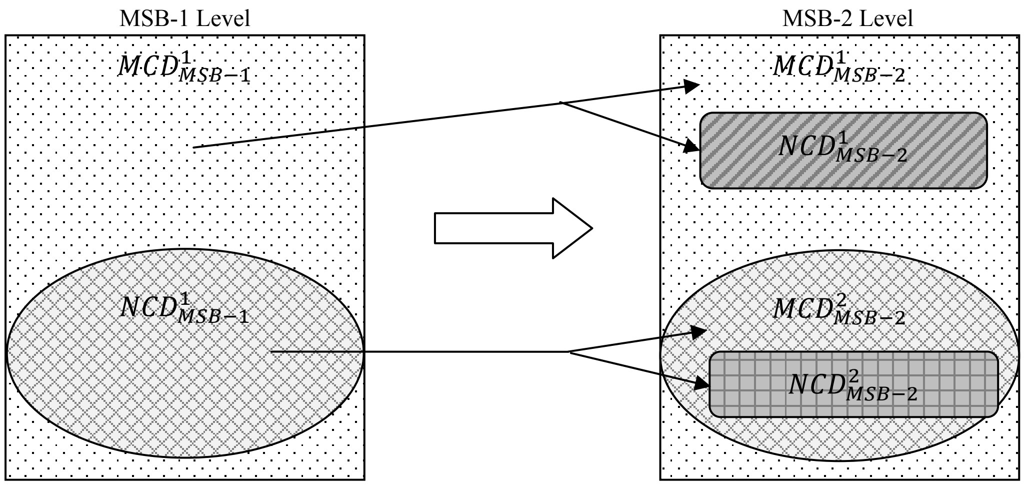


Fig. 2.4 Regions of discrimination for two levels.

Algorithm

Several steps that have been used in previous sections to compute the minimum or maximum of a data sequence are extended now to images. For simplicity, we assume that an erosion of the image is being performed.

Step 1: Compute the bitwise accumulation for the two branches resulting from one node of the tree in the precedent level. In this step, we just follow the structure of the tree depicted in Fig. 2.3. Note that $\mathcal{A}_{\ell+1}$ represents the bitwise accumulation of the node of the precedent level.

$$R_\ell^{\mathscr{b}} = \mathcal{A}_{\ell+1} \cup I_\ell \tag{37}$$

$$S_\ell^{\mathscr{b}} = \mathcal{A}_{\ell+2} \cup I_\ell \tag{38}$$

Step 2: Perform the binary erosion for each branch, which is similar to apply the Max or Min operator to a binary level of a data sequence.

$$\mathcal{E}_R = R_\ell^{\mathscr{b}} \ominus \mathrm{se} \tag{39}$$

$$\mathcal{E}_S = S_\ell^{\mathscr{b}} \ominus \mathrm{se} \tag{40}$$

Step 3: Extract the eroded segment of each branch through the masks that corresponds to the *MCD* and *NCD* cases.

$$\mathcal{R}_\ell^{b} = \mathcal{E}_R \cap MCD_\ell^{b} \tag{41}$$

$$\mathcal{S}_\ell^{b} = \mathcal{E}_s \cap NCD_\ell^{b} \tag{42}$$

Step 4: Get the boolean addition of the two cases of discrimination to yield a segment of the erosion that corresponds to a given node of the tree diagram.

$$i_\ell^{b} = \mathcal{R}_\ell^{b} \cup \mathcal{S}_\ell^{b} \tag{43}$$

Step 5: Compute the *NCD* and *MCD* regions for each of the resulting subsets. Note that $NCD_{\ell-1}^{1}$ and $NCD_{\ell-1}^{2}$ are the new regions where the *NCD* condition is established, and in accordance with Fig. 2.4, these new regions are subsets of the previous regions of upper levels. This happens because new information is introduced when *step 1* takes place, which redistributes the regions where the cases of candidate discrimination occur.

$$NCD_{\ell-1}^{b} = \mathcal{R}_\ell^{b} \tag{44}$$

$$NCD_{\ell-1}^{b+1} = \mathcal{S}_\ell^{b} \tag{45}$$

$$MCD_{\ell-1}^{b} = \overline{\mathcal{R}_\ell^{b}} \cap MCD_\ell^{b} \tag{46}$$

$$MCD_{\ell-1}^{b+1} = \overline{\mathcal{S}_\ell^{b}} \cap NCD_\ell^{b} \tag{47}$$

Finally, steps 1 to 5 are repeated in all the branches of a given level, the boolean addition of all partial erosions will yield the final erosion in a given bitplane (i.e. $I_\ell^t = i_\ell^{b} \cup i_\ell^{b+1} \ldots \cup i_\ell^{j}$), where $j = 2^{MSB-\ell}$. This procedure is extended to the subsequent bitplanes.

2.2.3 Proof of the Bitwise Erosion Algorithm

The algorithm of bitwise erosion proposed above is basically supported by the binary search minimization method, which is a well-known minimization technique. However its extension to images is not trivial and certain questions arise about its relation with the method of threshold decomposition. Therefore, such relationship should be investigated in order to shed some light about the theoretical insight that yields this new approach. For this purpose, let us demonstrate how the erosion obtained through the bitwise erosion algorithm yields the same mathematical expression as if the image were eroded through threshold decomposition technique. We will assume that the image has a dynamic range of 8, and the range comprised by using bitwise and threshold decomposition is presented in Fig. 2.5 and Fig. 2.6, respectively.

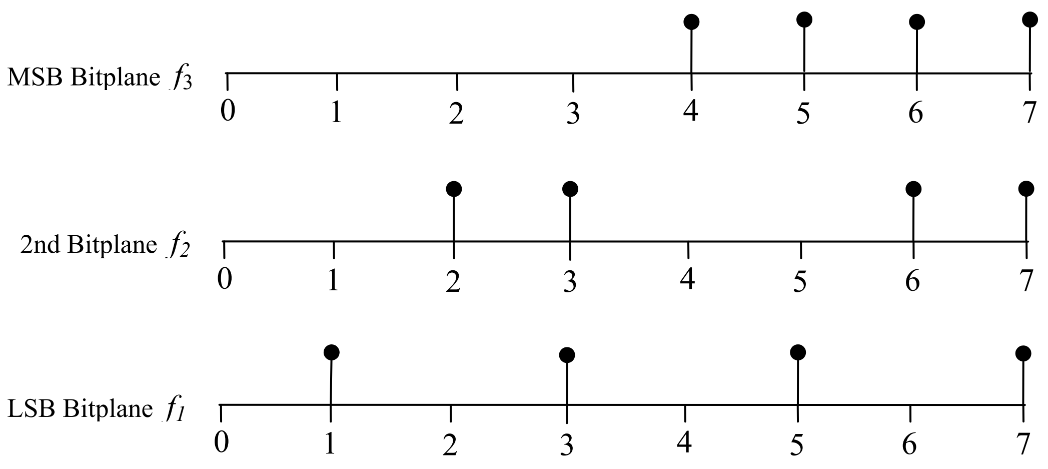


Fig. 2.5 Range comprised of every bitplane obtained by bitwise decomposition.

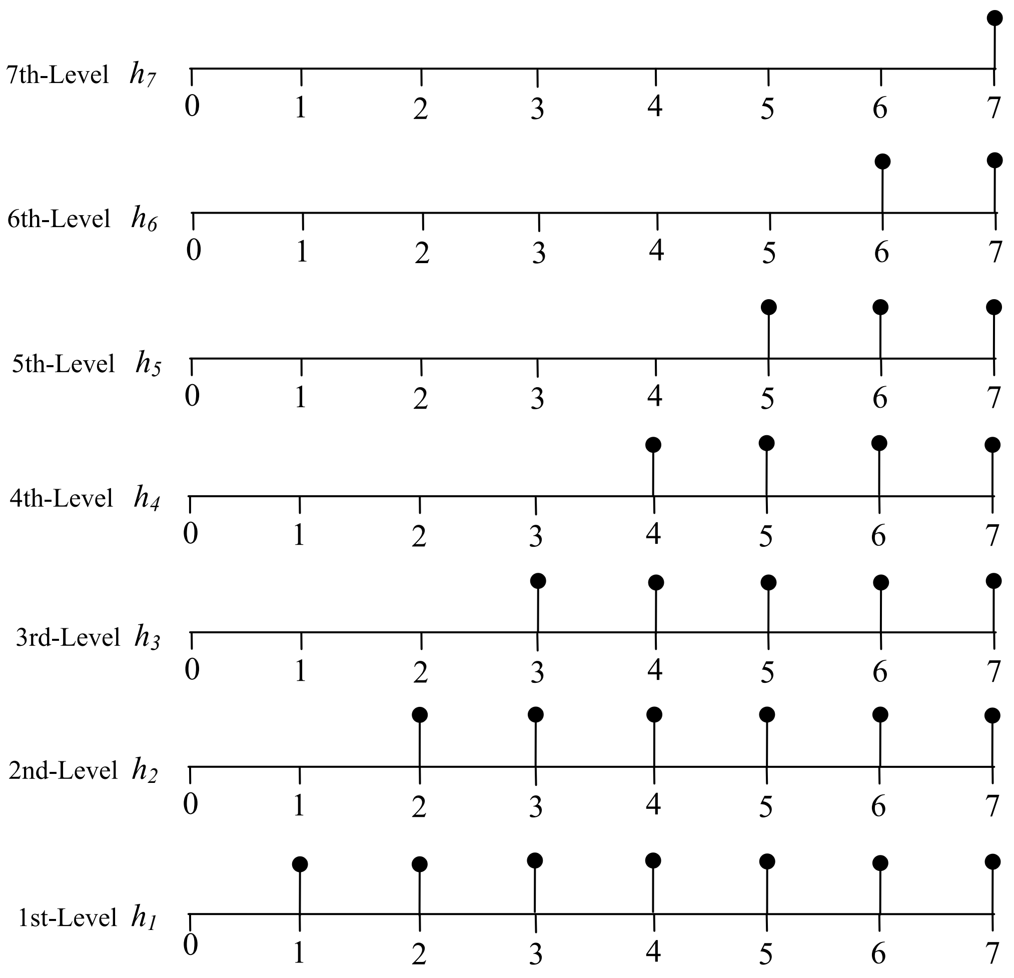


Fig. 2.6 Range comprised of every bitplane obtained by threshold decomposition.

Relationships Between Boolean and Real Number Algebra

Before to proceed to get the proof of the bitwise erosion algorithm, let introduce some properties that allow translate boolean to real number algebra. These properties can be very useful to ease the manipulation of boolean expressions and provide the framework to support the relationship between threshold decomposition and the erosion technique presented in this work. The following properties hold if and only if $g, g_1, g_2 \in \{0,1\}$.

Complement:

$$\bar{g} = 1 - g \tag{48}$$

Multiplicative:

$$g_1 \cap g_2 = g_1 g_2 \tag{49}$$

If $g_1 \subseteq g_2$:

$$g_1 \cap g_2 = g_1 \tag{50}$$

Distributive:

$$g_1 \cap \overline{g_2} = g_1(1 - g_2) = g_1 - g_1 g_2 \tag{51}$$

Additive:

$$\begin{aligned} g_1 \cup g_2 &= g_1 + g_2 - g_1 \cap g_2 \\ &= g_1 + g_2 - g_1 g_2 \end{aligned} \tag{52}$$

If two binary images are disjoint $g_1 \cap g_2 = \emptyset$, then:

$$g_1 \cup g_2 = g_1 + g_2 \tag{53}$$

Erosion of the MSB bitplane

The erosion of the MSB bitplane can be obtained straightforward, since there is no other biplane of greater significance. Also note from Fig. 2.5 and Fig. 2.6 that $f_3 = h_4$. Therefore, the erosion in terms of the bitwise and threshold decomposition is presented in (54).

$$I_3^t = f_3 \,\Theta\, se = h_4 \,\Theta\, se \tag{54}$$

where *se* is an arbitrary structuring element.

Erosion of the 2nd bitplane

In order to get the mathematical expression that yields the erosion of the 2nd bitplane, we will follow the steps explained above in equations (37)-(47). First, we state that the boolean addition of the 2nd bitplane with and without binary level rectification can be obtained as shown in (55) and (56).

$$R_2^1 = f_3 \cup f_2 \tag{55}$$

$$S_2^1 = f_2 \tag{56}$$

By following the steps depicted in eq. (39)-(42), we can reach (57) and (58). Observe that $MCD_2^1 = \overline{I_3^t}$ and $NCD_2^1 = I_3^t$, given that they represent the regions where the NCD and MCD cases were fulfilled in the past decision level. It is worth to point out that the erosion of the MSB plane tell us about the positions where the elements in the neighborhood were set to "1" (*NCD*). Therefore, the complement of this region automatically leads us to the *MCD* region.

$$\mathcal{R}_2^1 = (R_2^1 \,\Theta\, se) \cap \overline{I_3^t} \tag{57}$$

$$\mathcal{S}_2^1 = (S_2^1 \,\Theta\, se) \cap I_3^t \tag{58}$$

The erosion of the second bitplane is obtained according to (43) and rewritten in (59).

$$I_2^t = \left[(R_2^1 \,\Theta\, se) \cap \overline{I_3^t}\right] \cup \left[(S_2^1 \,\Theta\, se) \cap I_3^t\right] \tag{59}$$

The *NCD* and *MCD* regions of the next decision level can be computed by using eq. (44)-(47), yielding (60)-(63).

$$NCD_1^1 = \mathcal{R}_2^1 \tag{60}$$

$$NCD_1^2 = \mathcal{S}_2^1 \tag{61}$$

$$MCD_1^1 = \overline{\mathcal{R}_2^1} \cap MCD_2^1 \tag{62}$$

$$MCD_1^2 = \overline{\mathcal{S}_2^1} \cap NCD_2^1 \tag{63}$$

Now we will focus on express (59) in terms of the *h*-bitplanes, which represent the decomposition of the image by using TD property. Let rewrite (59) in its expanded form as stated in (64).

$$I_2^t = \left[\left((f_3 \cup f_2)\ \Theta\ se\right) \cap\ \overline{f_3\ \Theta\ se}\right] \cup \left[(f_2\ \Theta\ se) \cap (f_3\ \Theta\ se)\right] \tag{64}$$

We can start to simplify this equation by using the property that relates erosion with set operators; such property was previously established in (9), resulting (65).

$$I_2^t = \left[\left((f_3 \cup f_2)\ \Theta\ se\right) \cap\ \overline{f_3\ \Theta\ se}\right] \cup \left[(f_2 \cap f_3)\ \Theta\ se\ \right] \tag{65}$$

Note that in the present form of (65) is difficult to get a simpler expression in terms of set operators. Therefore, the relationships stated in (48)-(53) might help us to translate boolean to real number algebra. Then, applying (51) and (9) in (65) lead us to (66).

$$I_2^t = \left[(f_3 \cup f_2)\ \Theta\ se -\ \left((f_3 \cup f_2) \cap f_3\right)\ \Theta\ se\right] \cup \left[(f_2 \cap f_3)\ \Theta\ se\ \right] \tag{66}$$

Note that by helping us from Fig. 2.5 and the property stated in (50) and we can assert that $(f_3 \cup f_2) \cap f_3 = (f_3 \cap f_3) \cup (f_2 \cap f_3) = f_3$, yielding (67).

$$I_2^t = \left[(f_3 \cup f_2)\ \Theta\ se -\ f_3\ \Theta\ se\right] \cup \left[(f_2 \cap f_3)\ \Theta\ se\ \right] \tag{67}$$

Now, we can realize that in (67) the union set may be replaced by a sum applying (52), which lead us to (68).

$$I_2^t = (f_3 \cup f_2)\ \Theta\ se -\ f_3\ \Theta\ se + (f_2 \cap f_3)\ \Theta\ se \tag{68}$$

Finally, we can help us from Fig. 2.5 and Fig. 2.6 to express (68) in terms of *h*-bitplanes, since $f_3 \cup f_2 = h_2$, $f_3 = h_4$, $f_2 \cap f_3 = h_6$. Thus, we can reach (69).

$$I_2^t = h_2\ \Theta\ se + h_6\ \Theta\ se -\ h_4\ \Theta\ se \tag{69}$$

Erosion of the 3rd bitplane

If we follow the structure of the decision tree depicted in Fig. 2.3, we can see that in this level the algorithm processing is divided in two branches. Therefore each branch can be analyzed independently by considering its corresponding NCD and MDC cases and the bitplane accumulation from the 2nd bitplane. Thus, we proceed to get binary erosion corresponding to the 1st branch.

A similar procedure to the one performed in the 2nd bitplane is now followed for the 1st branch. Therefore, we directly present the erosion for each of the cases. Note that NCD_1^1 and MCD_1^1 were previously computed in (60) and (62).

$$\mathcal{R}_1^1 = [(f_3 \cup f_2 \cup f_1)\ \Theta\ se] \cap \left[\overline{\mathcal{R}_2^1} \cap \overline{I_3^t}\right] \tag{70}$$

$$\mathcal{S}_1^1 = [(f_3 \cup f_1)\ \Theta\ se] \cap\ \mathcal{R}_2^1 \tag{71}$$

In order to simplify (70) in terms of *h*-bitplanes, consider that $f_3 \cup f_2 \cup f_1 = h_1$ and the results obtained in the previous bitplane, which lead us to state (72) and finally assert (73).

$$\mathcal{R}_1^1 = (h_1\ \Theta\ se)(1 - h_2\ \Theta\ se +\ h_4\ \Theta\ se)(1 - h_4\ \Theta\ se) \tag{72}$$

$$\mathcal{R}_1^1 = h_1\ \Theta\ se - h_2\ \Theta\ se \tag{73}$$

On the other hand, (71) can be manipulated in a similar way as shown in (74).

$$\mathcal{S}_1^1 = [(f_3 \cup f_1)\ \Theta\ se][h_2\ \Theta\ se -\ h_4\ \Theta\ se] \tag{74}$$

Given that the property of multiplication is commutative we can recompose (74) as stated in (75).

$$\mathcal{S}_1^1 = \big((f_3 \cup f_1) \cap h_2\big)\ \Theta\ se - \big((f_3 \cup f_1) \cap h_4\big)\ \Theta\ se \tag{75}$$

By using simple algebra of sets we can get (76).

$$\mathcal{S}_1^1 = h_3\ \Theta\ se - h_4\ \Theta\ se \tag{76}$$

Finally, the erosion of the 1st branch can be obtained by substituting (73) and (76) in (77), which yields (78).

$$i_1^1 = \mathcal{R}_1^1 \cup \mathcal{S}_1^1 \tag{77}$$

$$i_1^1 = h_1\ \Theta\ se + h_3\ \Theta\ se - h_2\ \Theta\ se - h_4\ \Theta\ se \tag{78}$$

For the 2nd branch we can get

$$\mathcal{R}_1^2 = [(f_2 \cup f_1)\ \Theta\ se] \cap MCD_1^2 \tag{79}$$

$$\mathcal{S}_1^2 = (f_1\ \Theta\ se) \cap NCD_1^2 \tag{80}$$

Given that MCD_1^2 has been previously defined in (63), we can redefine (79) as in (81).

$$\mathcal{R}_1^2 = [(f_2 \cup f_1)\ \Theta\ se] \cap \left[\overline{\mathcal{S}_2^1} \cap I_3^t\right] \tag{81}$$

Consider that $(f_2 \cup f_1) \cap f_3 = h_5$ and $\mathcal{S}_2^1$ was derived in the 2nd bitplane, which yields (82) and its simplification (83).

$$\mathcal{R}_1^2 = (h_5\ \Theta\ se)(1 - h_6\ \Theta\ se) \tag{82}$$

$$\mathcal{R}_1^2 = h_5\ \Theta\ se - h_6\ \Theta\ se \tag{83}$$

On the other hand, we can simplify (80) by using (61) leading us to (84) and (85).

$$\mathcal{S}_1^2 = (f_1\ \Theta\ se)\ \cap \mathcal{S}_2^1 \tag{84}$$

$$\mathcal{S}_1^2 = (f_1 \cap h_6)\ \Theta\ se \tag{85}$$

From Fig. 2.5 and 2.6, we can state that $f_1 \cap h_6 = h_7$, yielding (86).

$$\mathcal{S}_1^2 = h_7\ \Theta\ se \tag{86}$$

It is interesting to observe that any subset of the bitplanes that correspond to one of the discrimination cases (i.e. $\mathcal{S}_\ell^{b}$, $\mathcal{R}_\ell^{b}$) can be regarded by a *difference of the eroded h-bitplanes*, which exactly correspond to the differences of translating *h*-bitplanes to bitwise bitplanes as you can observe in Fig. 2.5 and Fig. 2.6. Now, we can recompose the erosion of the 2nd branch in terms of the *h*-bitplanes as presented in (87). Given that the elements at both sides of the union set are disjoint we can use the property derived in (53), which results (88).

$$i_1^2 = (h_5\ \Theta\ se - h_6\ \Theta\ se) \cup (h_7\ \Theta\ se) \tag{87}$$

$$i_1^2 = h_7\ \Theta\ se + h_5\ \Theta\ se - h_6\ \Theta\ se \tag{88}$$

The boolean addition of the 1st and 2nd branches will finally lead us to the erosion of the 3rd bitplane as shown in (89).

$$I_1^t = i_1^1 \cup i_1^2 = h_1 \,\Theta\, se + h_3 \,\Theta\, se + h_5 \,\Theta\, se + h_7 \,\Theta\, se - \cdots$$
$$\cdots - h_2 \,\Theta\, se - h_4 \,\Theta\, se - h_6 \,\Theta\, se \tag{89}$$

Recomposition of the Bitwise Eroded Image

The eroded multilevel image can be recomposed from its bitplanes as shown in (90) and (91).

$$I_{er} = 2^2 * I_3^t + 2^1 * I_2^t + 2^0 * I_1^t \tag{90}$$

$$I_{er} = 4 * (h_4 \,\Theta\, se) + 2 * (h_2 \,\Theta\, se + h_6 \,\Theta\, se - h_4 \,\Theta\, se) + \cdots$$
$$\cdots + h_1 \,\Theta\, se + h_3 \,\Theta\, se + h_5 \,\Theta\, se + h_7 \,\Theta\, se - h_2 \,\Theta\, se - \cdots \tag{91}$$
$$\cdots - h_4 \,\Theta\, se - h_6 \,\Theta\, se$$

Finally, the simplification of (91) leads us to (92).

$$I_{er} = h_1 \,\Theta\, se + h_2 \,\Theta\, se + h_3 \,\Theta\, se + h_4 \,\Theta\, se + h_5 \,\Theta\, se + \cdots$$
$$\cdots + h_6 \,\Theta\, se + h_7 \,\Theta\, se \tag{92}$$

Note that the result is the same as if we were used the Threshold Decomposition technique, and this in turn not only proves that the Bitwise Erosion Algorithm is an erosion strictly speaking, but also provides significant theoretical insight about the relationship between monotone and non-monotone increasing image decomposition. As noted, bitwise decomposition is a highly compact binary representation of a multilevel signal in comparison to threshold decomposition, which involves intensive non-linear manipulation in order to unfold bitwise bitplanes into *h*-bitplanes. Also, it is worth to point out the fact that given the symmetry of the bitwise decomposition, the analysis presented here can be extended for greater number of bitplanes since every subset of a bitplane in certain level generates two subsets in the following one. As a result, every eroded subset will yield the difference of the eroded *h*-bitplanes that compose such subset. Another point that should be remarked is the importance of the properties presented in (48)-(53), since they are the key to link algebra of sets to real number algebra. Therefore, its application can be extended to other rank operators and enable their implementation under different block configurations and computer architectures, which may help to optimize their performance.

2.2.4 Bitwise Dilation and Multiple Transformations

The dilation of an image can be obtained analogously as we did with data sequences. Thus, two options can be followed: 1. Dilation by duality. 2. Transform the bitwise erosion algorithm to an algorithm that leads the dilation straightforward by using both duality property and the De Morgan's theorem. The former approach can be very suitable for applications where erosion and dilation operators are employed in the processing, since the same algorithm is used to compute both approaches and just a minimum overhead is necessary to obtain the dilation. The steps that should be followed to obtain the dilation are stated in the duality property presented in (7).

1. Get the one's complement of the bitplanes and obtain the reflection of the structuring element if this is not symmetric.

2. Apply the Bitwise Erosion Algorithm.

3. Get the one's complement of the transformed bitplanes.

As noted, the overhead introduced is a simple complement operator (NOT), which is applied in the input and output bitplanes.

On the other hand, if an application demands only dilation operator it would be more convenient to have an approach that might leads us to the dilation straightforward. By using the De Morgan's theorem (32) and (33), we can convert the bitwise erosion algorithm to actually a dilation transformation. Such transformation can be derived easily by replacing the union set operator $\cup$ by the intersection $\cap$, and the erosion symbol Θ by dilation $\oplus$, in the equations (37)-(47). Note that the complexity is the same as in the case of the of the bitwise erosion algorithm, but no overhead is introduced as in the case of dilation by duality.

Multiple erosion/dilation can be achieved by using the chain rules previously established in (16) and (17). If a priori we know that multiple morphological transformations will be applied to the image, we can transform each bitplane directly by dilating the structuring element in the number times that the image will be eroded. This new structuring element it is applied to the bitwise erosion algorithm yielding the corresponding erosion bitplanes as if they were obtained by consecutives erosions. Finally, opening and closing operators can be also derived from the bitwise erosion algorithm if erosion and dilation are applied in tandem.

Chapter 3

Results

"Science is but an image of the truth"

-Francis Bacon-

3.1 Bitplane Erosion

In order to illustrate how the bitwise erosion algorithm works at a bitplane level, let obtain the erosion of the bitplane decomposition of the *lena* image previously presented in Fig. 2.1. Moreover, we will relate the resulting eroded bitplanes with the equations obtained in the proof of the algorithm described in section 2.2.4. It is worth to mention that the equations derived are valid even though the dynamic range of *lena* image is not 8. The erosion is performed twice with a cross shape 4-connected structuring element.

The erosion of the most significant bitplane is depicted in Fig. 3.1(b). Recall that the bitwise erosion of the MSB bitplane is straightforward, thus we can observe how the feather in the hat and some other details in the background were removed respect to the original bitplane Fig. 3.1(a). Note also that all the regions set to "1" represent the NCD region of the following bitplane, whereas its complement represents the MCD.

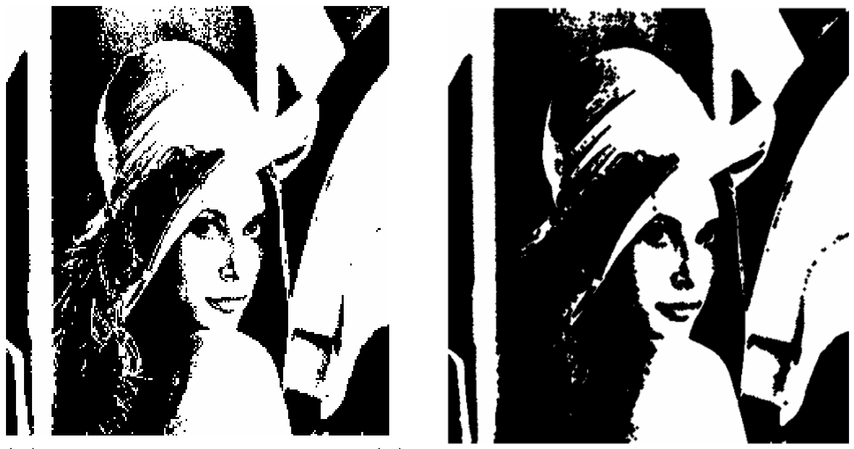


Fig. 3.1 (a) Original MSB bitplane (b) Bitwise erosion of the MSB bitplane.

For the second bitplane we know that the bitwise erosion will be splitted in two cases, each of them corresponds to the MCD and NCD conditions and processed according to equations (57) and (58). The resulting images are presented in Fig. 3.2(b) and Fig. 3.2(c).
Note that Fig. 3.2(b) is very close to the complement of Fig. 3.1(b), this is because the new information provided by the second bitplane is restricted to the MCD region, which is the complement of the erosion of the first bitplane. On the other hand, Fig. 3.2(c) shows the contribution of the second bitplane in the NCD region. At the end, in Fig. 3.3(a) the union set of the two regions is presented. Once again the unconnected pixels were removed and the edges of solid structures were reduced. However, some regions seem to be dilated instead when we compare the upper part of Fig. 3.3(a) and the original bitplane presented in Fig. 3.2(a). This happens because the contribution of the first plane fills some of the holes of the second bitplane, thus some regions look as if they were dilated.

In order to contrast the effect of applying directly the erosion operator to a bitplane without considering discriminative cases let introduce Fig. 3.3(b). On the other hand, the error generated by these two approaches it is depicted in Fig. 3.3(c). Note that the differences are significant and the main errors lay on the contours as well as regions where the contribution of the first bitplane is considerable; that is, the regions that looked dilated (i.e. top of the hat, upper regions of the bitplane). This example proves that TD property does not hold under a bitwise decomposition and the errors in the resulting gray-scale image might become very noticeable, especially in the contours. It is worth to remark that the binary images in Fig. 3.2 (b)-(c) will be used to compute the discriminative cases in the following decision level.

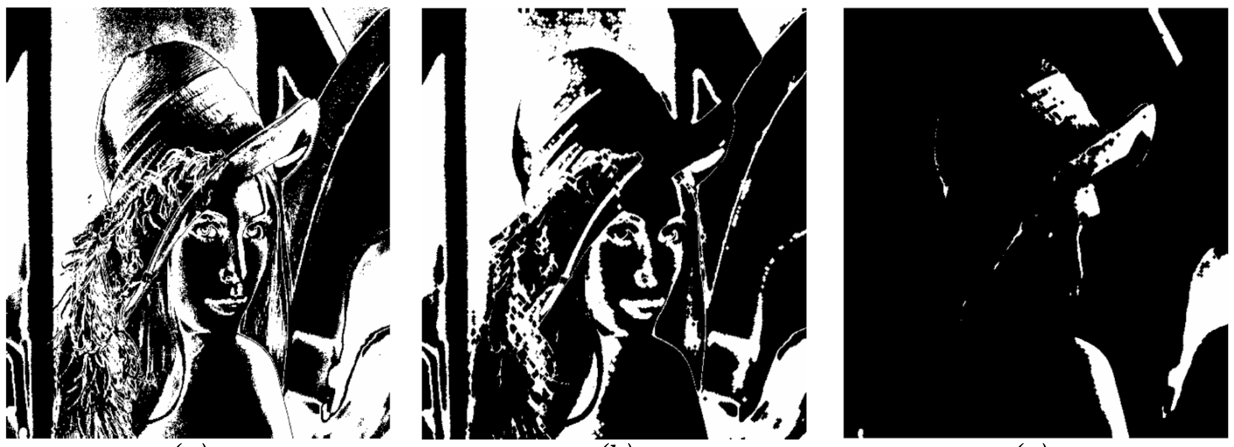


Fig. 3.2 (a) Original 2nd bitplane. (b) Erosion of the MCD branch ($\mathcal{R}_2^1$) (c) Erosion of the NCD branch ($\mathcal{S}_2^1$).

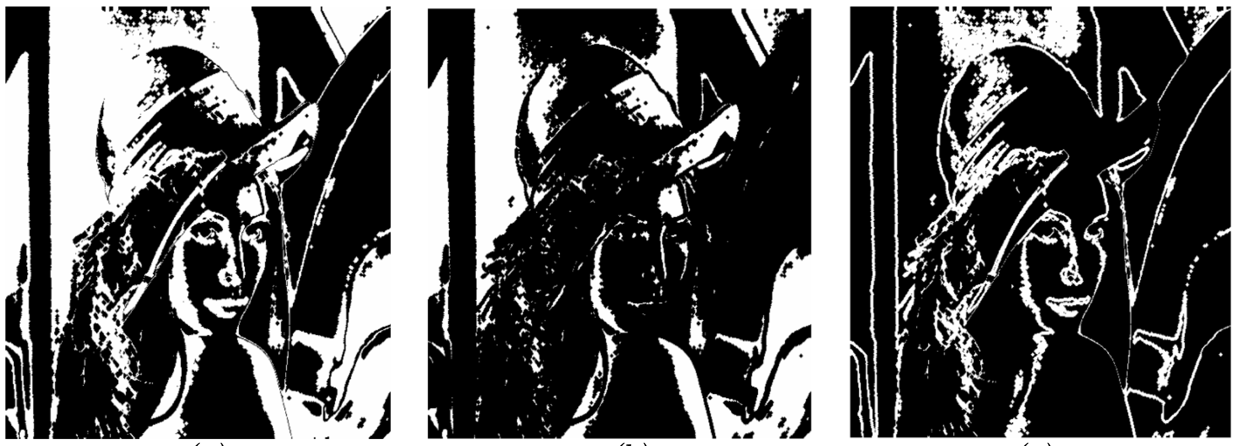

(a) (b) (c)

Fig. 3.3 (a) Eroded 2nd bitplane (I_2^t) (b) Erosion without discriminative conditions (c) Error.

For the third bitplane, every discriminative region in the previous decision level generates an independent branch in the current level, which we have called *1st* and *2nd* in the mathematical development performed in section 2.2.4. Fig. 3.4(b) and Fig. 3.4(c) shows the erosion that correspond to the MCD and NCD conditions for the *1*-branch, respectively. On the other hand, Fig. 3.4(d) and Fig. 3.4(e) represent the same conditions but for the *2*-branch. The original bitplane presented in Fig. 3.4(a) may be contrasted with the resulting bitwise erosion in Fig. 3.4(f).

It is interesting to observe that the resulting union set of all the discrimination cases returns a bitplane that looks dilated rather than eroded. Once again, this effect is result of the boolean addition of the precedent bitplanes; nevertheless, we can see some regions at the right side of the bitplane that were evidently eroded. Note that the bitplane in Fig, 3.4(e) is almost empty, which means that we are very certain about the minimum state information of the bitplane derived from the 2nd branch. In other words, it is very likely that at certain level of processing, some bitplanes of the discrimination cases will be completely filled with zeros; this implies that the binary search minimization technique has found the minimum in the entire region of discrimination and no exhaustive search should be continued.

Identifying when empty bitplanes occur, help to speed the algorithm up and save resources. Since every bitplane generates two bitplanes in the following levels and continues successively in this way, empty bitplanes help to improve the execution time as those appear in the upper levels of the tree. The number of empty bitplanes depends on the histogram distribution of the image. That is, if the image has a low dynamic range and the modes of the histogram have low variance, it is very probable that the number of empty bitplanes will be considerable and the erosion will be achieved faster. Conversely, if the histogram tends to be uniform distributed along the whole dynamic range, exhaustive search is required to reach the entire erosion, and it is complexity might be comparable to the threshold decomposition approach in terms of number of bitplane erosions.

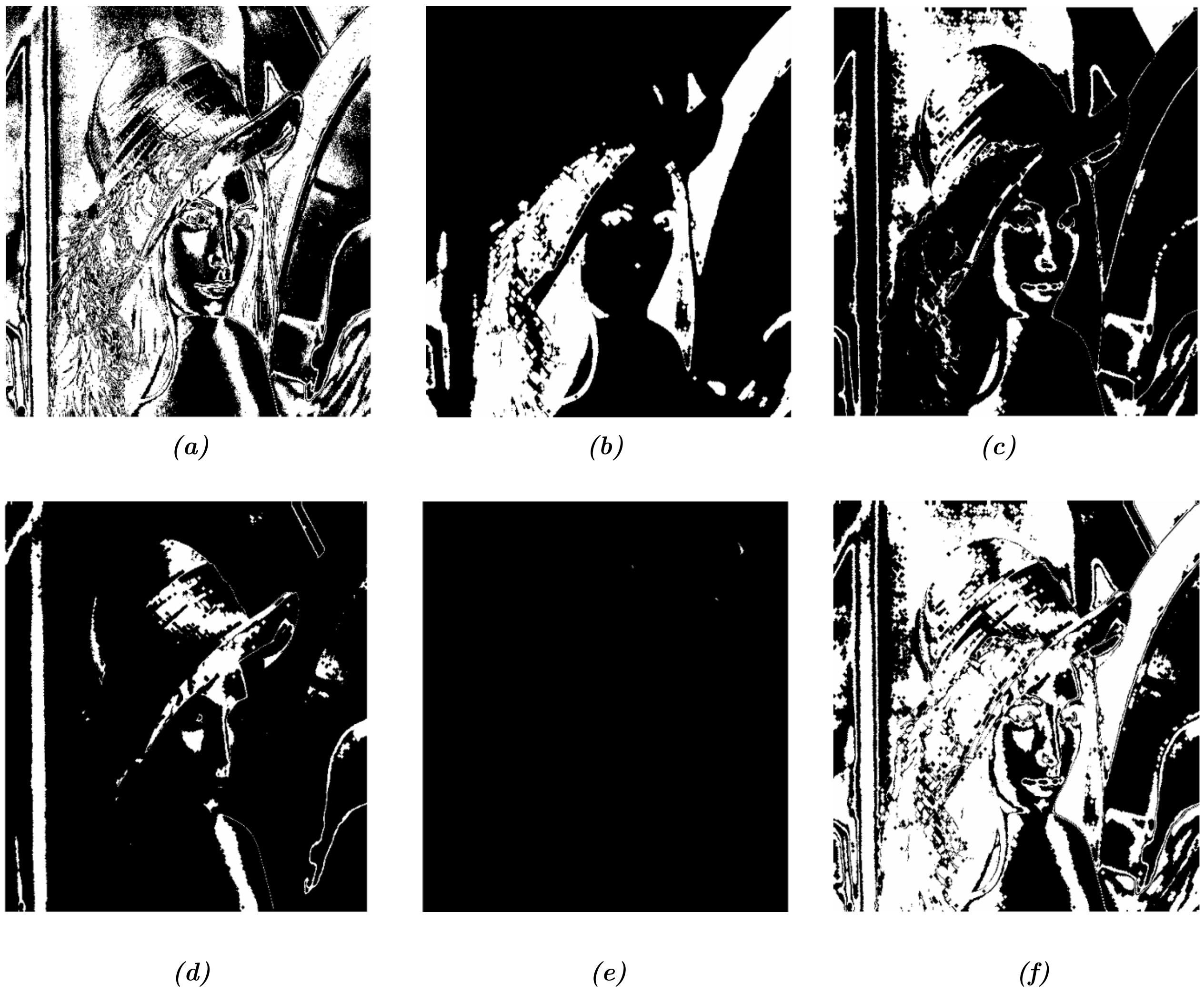


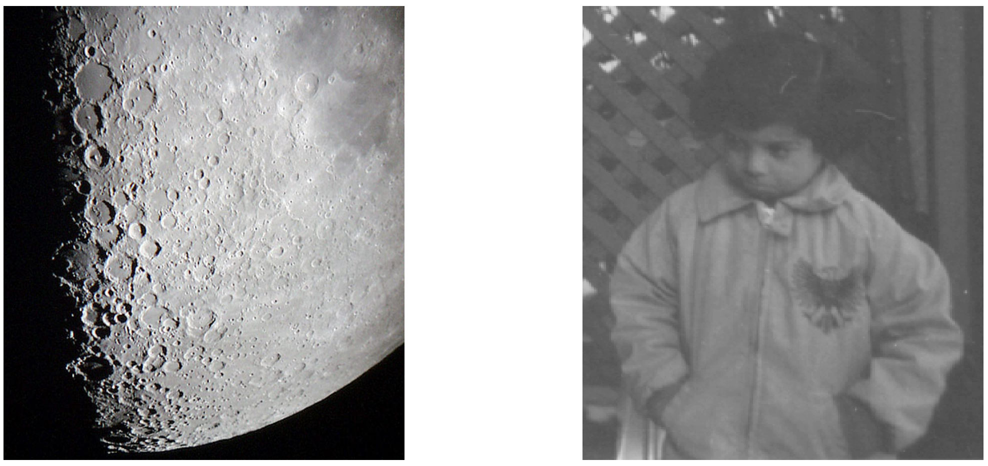
Fig. 3.4 (a) Original 3rd bitplane. Erosion of the (b) MCD region of 1-branch (c) NCD region of the 1-branch (d) MCD region of 2-branch (e) NCD region of the 2-branch (f) Bitwise erosion of the 3rd bitplane.

Normally, natural images do exhibit neither a high dynamic range nor a uniform distribution, which supports the convenience of our approach in relation to TD. In this sense the present approach reveals some entropic properties, since the computation time is greater when the entropy is maximum (uniform distributed) than when is tightly concentrated in some gray-scale range. These properties will be discussed in detail in the following section.

Finally, the transformation of subsequent bitplanes is performed following the steps of the bitwise erosion algorithm and the structure of the level tree diagram till reach the last bitplane.

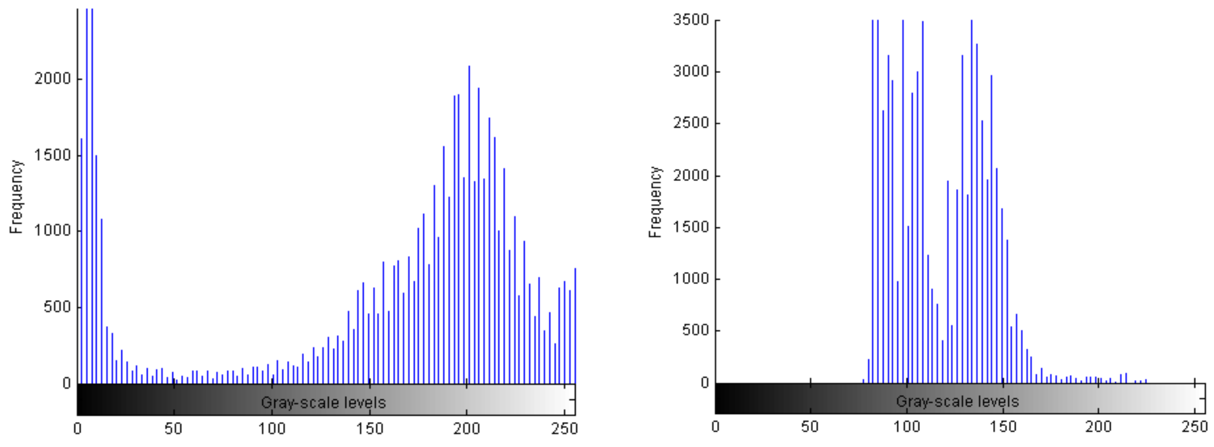
3.2 Entropic Properties of the Bitwise Erosion Algorithm

As discussed in previous section, the histogram of the image to be processed with the Bitwise Erosion Algorithm plays a key role in the processing time. Let us introduce some examples in order to show the performance of the algorithm in two opposed circumstances.
Fig. 3.5(a) shows an image of the moon with a high dynamic range, while Fig. 3.5(b) shows a test image with low dynamic range. The respective histograms are depicted in Fig. 3.6(a) and Fig. 3.6(b).



(a) *(b)*

Fig. 3.5 (a) High (b) Low dynamic range image.



(a) *(b)*

Fig. 3.6 Histogram of the (a) Moon (b) Girl.

The algorithm was run in Matlab to test the processing time of these two images with the same resolution 256x256. In Table 3.1, the processing times of the two images under the BEA and TD approaches is presented. Note that the processing time under bitwise erosion algorithm is much lower than threshold decomposition, even though the algorithms are not implemented in real time. There are several reasons by which BEA outstands: 1. No excessive overhead is introduced when the gray-scale image is decomposed and recomposed. 2. The algorithm is able to determine when it is not necessary to carry out exhaustive search, which avoids extra computation in subsequent decision levels. 3. Bitplanes are packed and processed as a one-dimensional array with elements that have a 32-bit unsigned integer format. This sort of packing is usually employed to speed binary morphological operators, and allow manipulation at bitwise level. This last feature enables us to process the entire algorithm in a packed form and only unpack the bitplanes resulting of the transformation. This is not possible under a TD approach given that as a bitplane is transformed it is added to the previous bitplanes, and such a numeric addition is not supported when the bitplane is packed.

On the other hand, the number of binary erosions is a reliable measure that tells us how the algorithm performs independently on the computer architecture in which is implemented. This is measure is based on the fact that the binary erosion of a bitplane is the main consuming time processing in both approaches. Also, the complexity in terms of number of erosions is the same when the images have maximum entropy (i.e. 255 erosions for an 8-bit image). Note that the erosion of the girl under BEA approach was performed faster than the moon image thanks to the entropic properties of the algorithm. As a result, it was necessary only 155 erosions, while in the image of the moon 86 extra erosions are required. Moreover, the images treated with the BEA show some reduction in the number of erosions in comparison to the images eroded with TD.

It is important to remark that in order to speed up the processing time of the TD approach when the image has low dynamic range (*e.g.* Image of the girl), it is mandatory to know the maximum and minimum of the image to reduce the number of erosions, while in the BEA it is not necessary to know a priori this information.

Finally, in the last column of Table 3.1 the percentage of saved time for each approach is presented. This percentage is measured in relation to the maximum time spent to compute the erosion of a uniform distributed image under TD approach. Observe that BEA leads to significant saving time, which becomes more evident when the resolution of the image increases.

Table 3.1 Assessment of the entropic properties of BEA and TD approaches.

	Bitwise Erosion Algorithm		Threshold Decomposition	
	Girl	Moon	Girl	Moon
Time	0.428s	0.734s	1.422s	2.422s
No. of Erosions	155	241	178	255
% of Saving time	82.33	70	41.3	0

3.2.1 Speed and Quantization Tradeoff

As discussed previously, the speed of the BEA strongly depends on the image distribution. Therefore, it would be interesting to know the performance of the algorithm when the histogram is manipulated to get better processing times. A logical next step is to linearly map the image into the dynamic range of b bits. The intention is to avoid processing bitplanes as the bit quantization decreases, and check the error produced by such quantization. In Fig. 3.7 we can observe how the quality of the eroded image degrades as the bit quantization decreases. However, it should be also mentioned that from the qualitative point of view, the error or distortion produced by the quantization is not noticeable at first glance for Fig. 3.7(a)-(c). Therefore, if what matters after performing the erosion is the visual effect produced, it would be convenient to quantize the image with a lower precision. This approach would be very helpful when the image has high resolution and the erosion has to be performed in real time.

Note that this sort of manipulation yields a tradeoff between processing speed and quality of the eroded image. In some sense, this tradeoff resembles the one existing in image compression. However, in this case, the aim is not reduce the image data size, but the processing time.

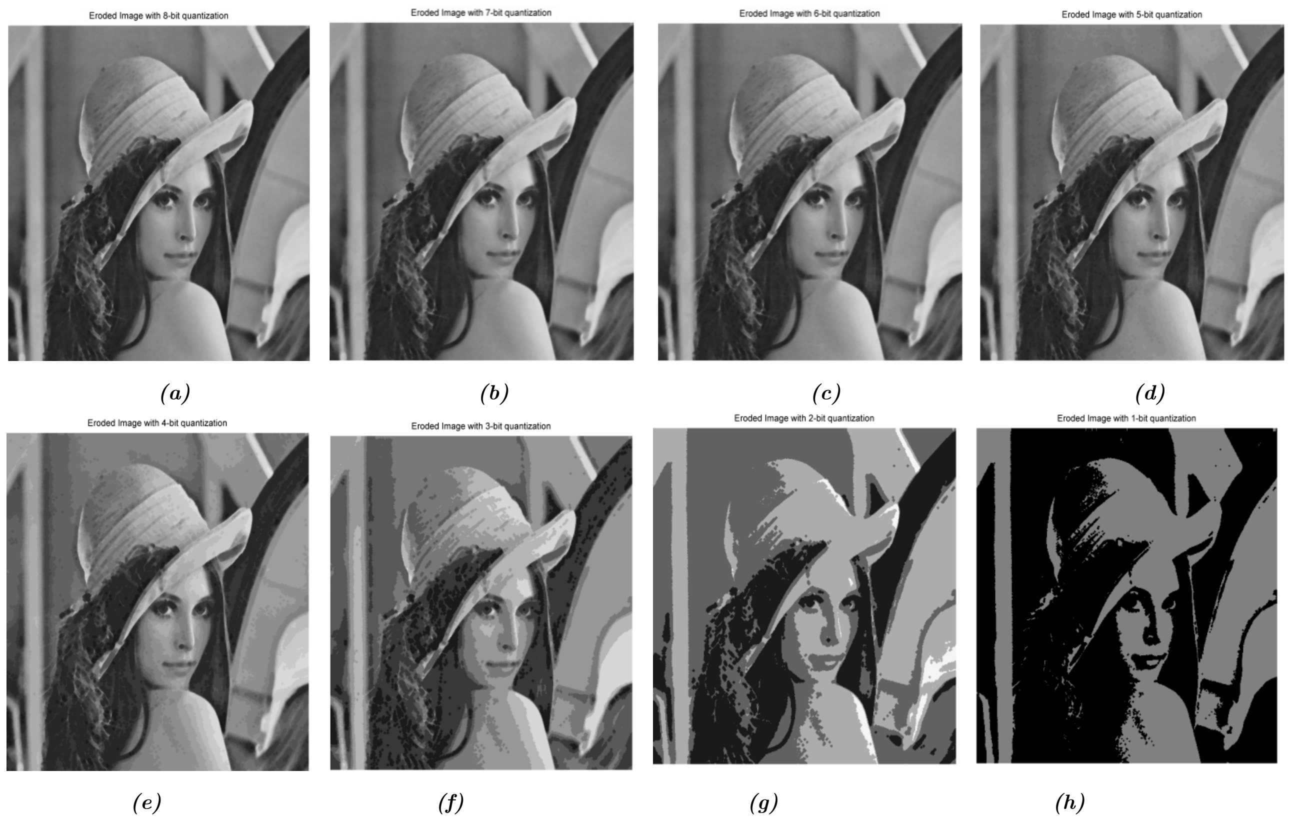

Eroded Image with 8-bit quantization
Eroded Image with 7-bit quantization
Eroded Image with 6-bit quantization
Eroded Image with 5-bit quantization
(a)
(b)
(c)
(d)
Eroded Image with 4-bit quantization
Eroded Image with 3-bit quantization
Eroded Image with 2-bit quantization
Eroded Image with 1-bit quantization
(e)
(f)
(g)
(h)

Fig. 3.7 Bitwise erosion for several levels of bit quantization.

In order to make more evident the tradeoff between the processing speed and quantization we present in Fig. 3.8 the execution time and the PSNR for several bits of quantization. In Fig. 3.8(a) we can observe that the execution time decreases exponentially as the number of quantization levels is reduced. The improvement in the processing time is very significant even when the precision has been reduced in 1 bit. It is worth to mention that the timings presented here were obtained considering that the image has a 512x512 pixel resolution running under Matlab. At the moment, the aim is only to show the behavior of the algorithm; however, these figures become smaller either when implemented in a lower level language or under the Eye-RIS architecture.

On the other hand, in Fig. 3.8(b) we can assess the degradation suffered by reducing the image precision. Fortunately, the loss of the image quality does not decay exponentially, but nearly in a uniform way. Considering the logarithmic response of the eye to intensity variations, the behavior of the curve justifies applying some bit quantization without compromising too much the quality of the image and speeding up the processing time.

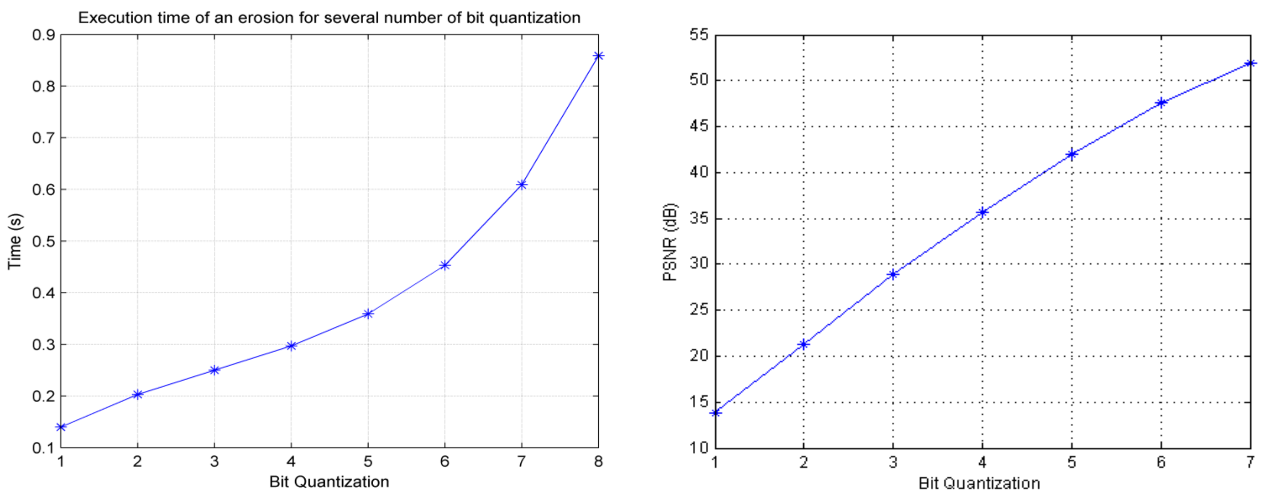


Fig. 3.8 (a) Execution time (b) PSNR, for several levels of bit quantization.

In Fig. 3.7 we observed in some cases that even with 5-bit quantization, the erode image seem to be free of the visual quantization effects, which tell us that the erosion could be carried out in 0.36s rather than 0.859s. In several tests, we have found that usually 5 or 6 bit quantization is enough to get the erosion of an image without compromising too much its quality. This would imply a saving time of around 50% of the total time required to process the image with 8-bit quantization.

A non-linear mapping applied to the histogram that intends to improve the processing speed might yield very bad results in terms of the image quality. This means that, in addition to the quantization effects, some errors are introduced due to the alteration of the rank order in the neighborhood of the structuring element, which leads us to get some parts of the image that are eroded incorrectly. As a result, the PSNR might decay exponentially rather than uniformly.

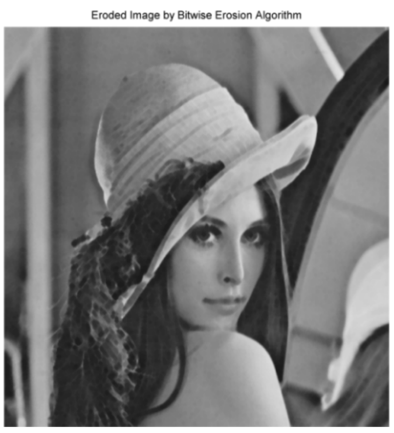
The entropic properties of the bitwise erosion algorithm presented in this section might have significant repercussions in applications such as visual prosthetics where morphological operators are largely employed to extract edges and shapes. Actual algorithms to compute morphological operations are rather complicated to be embedded in a visual implant given the reduced chip area and might not perform in real time. Recently, a somewhat similar approach to the BEA was proposed by Jabakhanji and Shams [18] to be used for detecting edges in visual implants. However, their proposal stands on performing the straight erosion of the binary planes decomposed in a bitwise fashion. As we know, strictly speaking such treatment does not yield the correct erosion, since the bitwise decomposition is not monotone increasing. Therefore, applying this approach might yield inaccurate results. Conversely, in the previous chapter we proved that bitwise erosion algorithm leads to formal erosion, and along this section we have shown that the algorithm can tradeoff its complexity with some loss in the image quality, which can be estimated. As a result, the BEA provides a solid framework that can be used to deal with morphological operators in reduced chip areas such as visual implants. Finally, we may conclude that the image distribution is an important feature that determines the performance of the BEA. The entropic properties envision several applications in which the algorithm can be very suitable. On the other hand, we show that the threshold decomposition approach demands computation requirements that overpass the complexity of diverse embedded applications and introduces considerable overhead when the image is being decomposed and recomposed.

3.3 Morphological Rank Spectrum

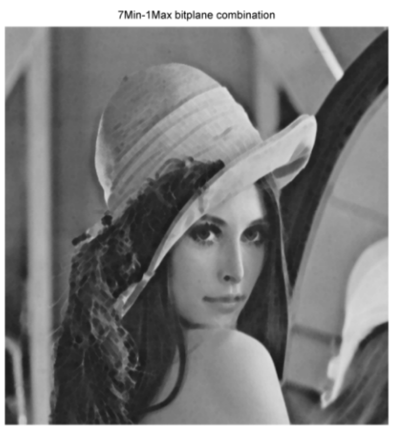
In this section will be presented some features of the bitwise erosion algorithm that are generated thanks to flexibility of the algorithm. A wide range of morphological operators are possible to be implemented by slightly modifying the BEA. As we know, the bitwise erosion algorithm is based on the binary search minimization, which is not only able to find the minimum value in a finite sequence, but also regard any element of such sequence. Minimum changes have to be done to the BEA to perform such a rank searching as we did in section 2.2, but for a bitplane level. That is, we have to set the *Max* or *Min* operator on each bitplane to regard the desired value. In our case, we are interested in investigating the effect produced when the bitplanes are neither eroded nor dilated in all the bitplanes. Recall that in order to obtain an erosion, the combination of rank operators that yield such morphological filter is (*Min-Min-Min-Min-Min-Min-Min-Min)* for an image with a bitdepth of 8. This means that in all the bitplanes in which the image is splitted, a minimization is carried out. Moreover, the position of every operator is linked to the hierarchical order of each bitplane. Conversely, for a dilation the combination is (*Max-Max-Max-Max-Max-Max-Max-Max).*

As a result, there exist 256 different combinations that can be used. As a matter of general interest, it would be convenient to look for a set of combinations that might lead us from an erosion to a dilation. In other words, the spectrum of morphological transformations that lies in between. In order to analyze such effect, we decided to progressively change the *Min* to *Max* operator from the least to most significant bitplane. To simplify the addressing of each combination, we established the following convention *n Min - m Max*, where *n* is the number of the most significant bitplanes whose operator is a minimum, whereas *m* is the number of the remaining bitplanes whose operator is a maximum. Note that the order of the Max/Min operators in this notation can be switched accordingly in order to get another set of rank combinations.

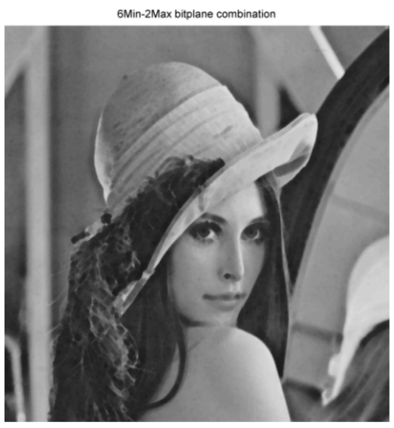
In Fig. 3.9 we can observe the result of applying several combinations that go from the erosion to dilation. Note that the bit planes are regarded in the form *n Min – m Max*, which means that a minimization is performed in the most significant bitplanes, and thus an erosion effect will be produced in the resulting image. When the less significative bitplanes are set to a maximum operator (i.e. Figs. 3.9 (b)-(c)) there is not significant changes respect to Fig. 3.9(a), which represents full erosion. On the other hand, in Figs. 3.9(d)-(f) we can start to see some variations that suggest that white (maxima) areas are being enhanced. Finally, in Fig. 3.9(g)-(h) show that they are more related to Fig. 3.9(i), which represents the combination that yields a full dilation. Note also, that in these last cases most of the bitplanes are set to *Max*.



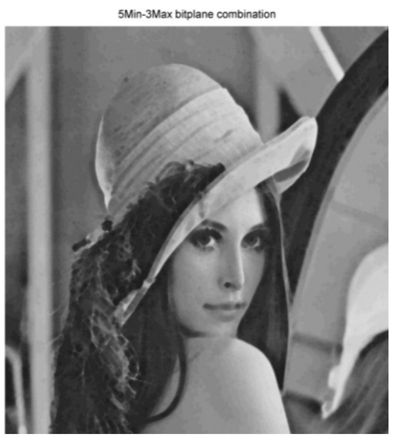
(a)



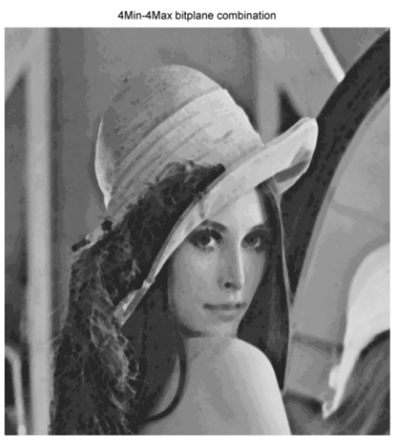
(b)



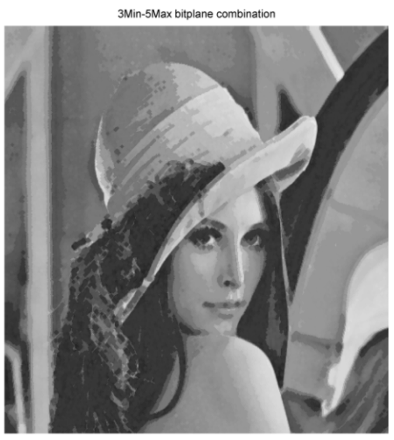
(c)



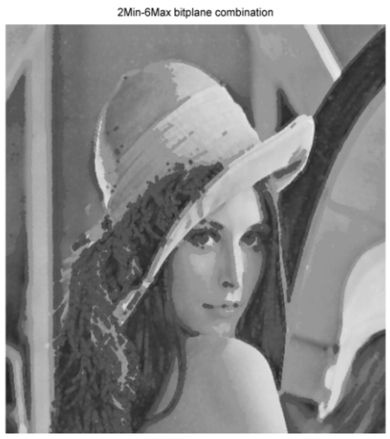
(d)

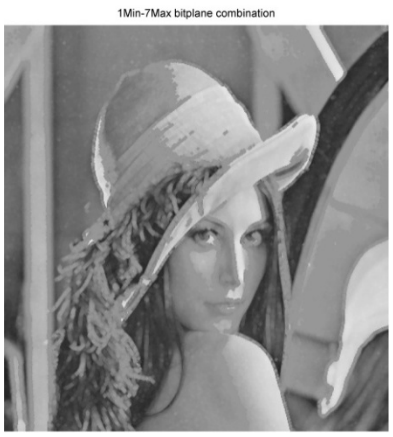


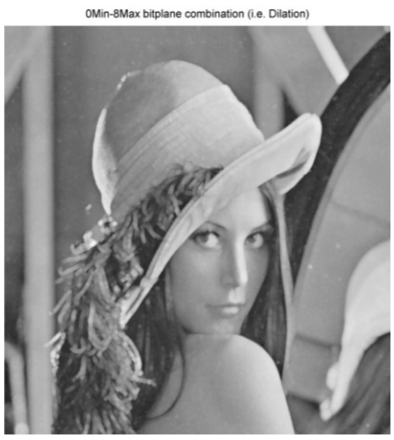
(e)



(f)



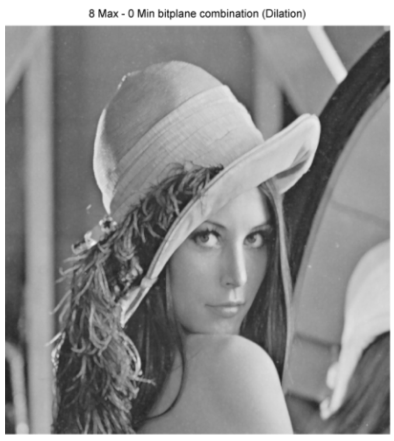


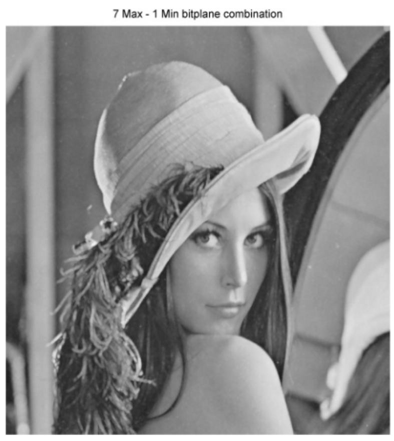


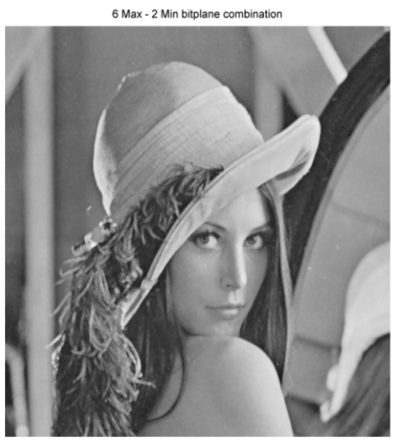
(g) *(h)* *(i)*

Fig. 3.9 Morphological transformations by using several Min/Max bitplane combinations.

On the other hand, it would be also interesting to know how we can reach progressively an erosion from a dilation. In this case, we tried the combinations where the maximum is applied on the most significant bitplanes, that is, we are considering the notation *m Max - n Min*. As a result, most of the resulting images will be more alike to a dilation as you can see in Figs. 3.10(b)-(c). Note that in Figs. 3.10(d)-(f) dark areas tend to be enhanced progressively, which proves that the sequence combinations tend from dilation to erosion as expected. Finally, in Figs. 3.10(g)-(i) we can observe that the combinations in which most of the bitplanes are set to a minimum operator, the erosion effect was more noticeable. Most of the changes become more evident in the hat, feathers, and background of the image.







(a) *(b)* *(c)*

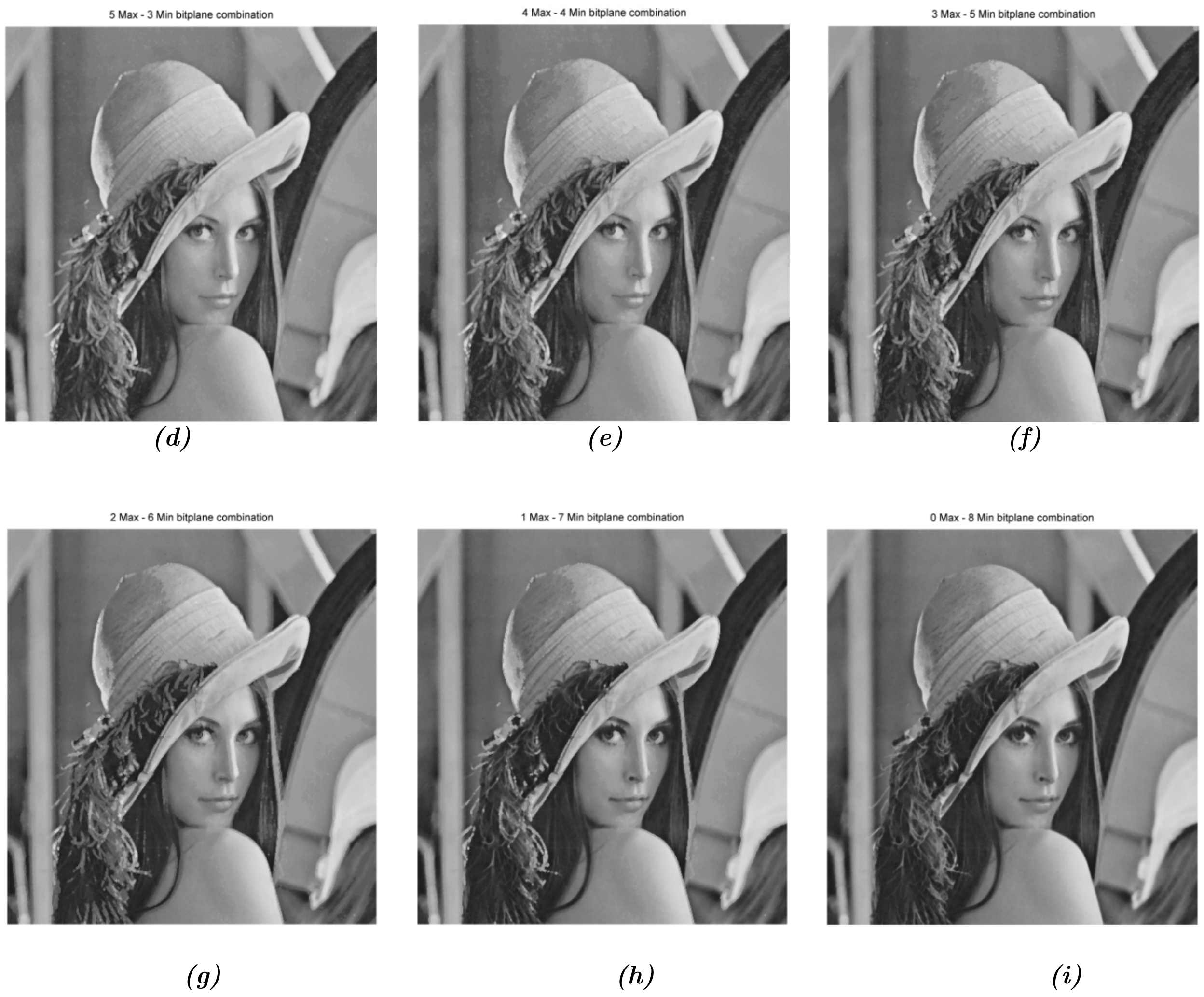

5 Max - 3 Min bitplane combination
4 Max - 4 Min bitplane combination
3 Max - 5 Min bitplane combination
(d)
(e)
(f)
2 Max - 6 Min bitplane combination
1 Max - 7 Min bitplane combination
0 Max - 8 Min bitplane combination
(g)
(h)
(i)

Fig. 3.10 Morphological transformations by using several Max-Min bitplane combinations.

In Fig. 3.9 and 3.10 we have seen the morphological transformations that exist between an erosion and dilation when the transformation is performed at a bitplane level. We have noted that the combinations that lead us from an erosion to dilation and vice versa depend strongly on the operator applied to the most significant bitplanes. This fact explains why the transition is performed in a logarithm fashion.

It is worth to remark that the bitwise erosion algorithm provides significant versatility in terms of the number of applications in which can be used. Also, note that the overhead introduced to obtain such a morphological spectrum is minimum, since the duality property enables us to perform erosion and dilation with the same BEA implementation.

Finally, ongoing research on such morphological transformation is being performed to unveil the relationship with other well-known non-linear filters such as the median and range filters, while some attempts to establish a formal mathematical framework are also being developed.

3.4 Algorithm Complexity

As mentioned previously, one of the main advantages of the bitwise erosion algorithm over threshold decomposition is the low overhead introduced when the multilevel image is decomposed and recomposed. This assertion can be supported by a complexity analysis as the one provided below. Such analysis is performed considering that the image occupies the full dynamic range, 8 bit depth, and n x n pixel resolution.

Firstly, let us derive the complexity of performing an image erosion by using the threshold decomposition approach. As we know, the algorithm is described in three steps:

1. Decomposition of the gray-scale into bitplanes by using TD property. This step requires an array of n x n digital comparators of 8 bits. According to the logic diagram of a typical comparator such as sn74ls518 [19], each device is composed in terms of logic gates as stated in (93).

$$N_{comp} = 8\,XNOR + 7AND \tag{93}$$

Also, we know that 1 XNOR = 4 NAND, and n^2 comparators are necessary to derive in parallel one bitplane. Therefore, the total hardware required is presented in (94).

$$N_{comp} = 39\,n^2\ gates \tag{94}$$

2. The erosion is applied to each bitplane that results from the threshold decomposition. Since we are assuming that the image is full in range, 2^8-1 bitplanes will be generated from the decomposition and thus the number of binary erosions. Given that there exist several methods to implement binary erosion and it is hard to say how many logic gates are necessary for each approach. Therefore, it is more convenient keep our analysis simple by only considering the number of erosions required. As a result in (94) we state that we need 255 bitplane erosions.

$$N_{erosion} = 255 \tag{95}$$

3. The recomposition of the eroded image is obtained by stacking. That is, the numeric addition of all the bitplanes is performed. Consequently, n^2 full adders of 8 bits are necessary to perform the stacking in parallel at a bitplane level. Normally, a full adder of 8 bits is composed in terms of logic gates as presented in (96).

$$N_{add} = 16\,XOR + 16\,AND + 8\,OR = 88\,gates \tag{96}$$

Recall that 1 XNOR = 4 NAND and the complexity of a OR gate is equal to a AND gate. Finally, the numbers of adders depend on the resolution of the image, which lead us to (97).

$$N_{stack} = 88\,n^2\ gates \tag{97}$$

Now, we will focus on analyze the complexity of the bitwise erosion algorithm. Unlike to the TD approach, in the bitwise erosion algorithm no further processing is required than reading/writing on the data of the bit fields when the image is decomposed and recomposed. However, the algorithm requires some bitwise manipulation at a bitplane level to carry out an erosion. In past chapters we have described the BEA through the equations (37) - (47). We can easily realize that steps 1, 3, 4, 5 only require two kind of operators, OR and AND gates, which replace the union ($\cup$) and intersection ($\cap$) operators, respectively. As a result, the hardware required to accomplish such operations at bitplane level is presented in (98).

$$N_{set} = 2n^2\ gates \tag{98}$$

On the other hand, in step 2 of the algorithm is shown that single erosion is performed by each region of the discriminative cases established by the level decision tree. The number of regions grows by a factor of 2 from one level to other. Based on this, we can say that the complexity in terms of binary erosions of the BEA is the same as the TD, as demonstrated in (99).

$$N_{erosion} = \sum_{i=0}^{7} 2^i = 2^8 - 1 = 255 \tag{99}$$

It is important to remark that this figure only takes place when the histogram is roughly uniform distributed along the whole dynamic range. Otherwise, this number may be significant reduced given the entropic properties of the BEA.

Assessing the hardware complexity of the two erosion approaches, we can readily conclude that BEA reduces significantly the hardware requirements and that the bitplane decomposition and stacking represent a serious burden for the TD approach as exposed previously.

A summary of the complexity of the TD and BEA algorithm is presented in Table 3.2. The hardware complexity was computed based on the basic building blocks that construct the algorithm. However, the complexity also depends on the number of times in which such blocks have been called. As a result, it is convenient to analyze how the number of calls to functions might affect the time processing of every approach.

The last column of Table 3.2 reveals important information about the performance of every approach. On the one hand, for the TD we can see that number of calls for the decomposition and stacking agrees with the number of bitplanes, which implies that these variables are linearly related at a cost of bulky hardware. On the other hand, the BEA seems to behave exponentially in relation to the number of bitplanes, but the elemental building block is simpler in terms of hardware, which suggests a faster execution and no latencies. Finally, we may confirm that BEA leads to a faster and chip-area-efficient algorithm to carry out basic morphological filters.

Table 3.2 Comparison of the complexity of TD and BEA algorithm.

	TD			***BEA***	
	Decomp.	*Erosion*	*Stacking*	*Sets*	*Erosion*
No. Gates	39 n^2	N.A.	88 n^2	2 n^2	N.A.
Calls	255	255	255	1024	255

3.5 Performance of the BEA in the Eye-RIS vision system.

As pointed out in Chapter 1, an important characteristic of the Eye-RIS vision system is that all the pixels on the image are processed in parallel and in real time. Such parallelism is fully exploited by the BEA since binary erosions and bitwise manipulation are carried out by the Q-Eye (analog processor), which performs far better than any digital counterpart. Despite of the outstanding features of the analog processor, the BEA cannot be implemented thoroughly in the Q-Eye given the limited precision when a gray-scale image is being manipulated. Particularly, in the decomposition/recomposition of the gray-scale image it is necessary manipulate the image with an accuracy of 1 LSB, which becomes an issue for the Q-Eye due to the noise introduced. Furthermore, the decomposition and recomposition of an image seems to be a task more suitable for the digital processor, since the actual pixel data is stored in a bitwise fashion. By appointing each task of the algorithm to the corresponding processor, we guarantee that our approach is implemented efficiently and in accordance with architecture available.

In previous sections we have defined that the BEA as an entropic-based erosion algorithm, which means that the processing time depends strongly on the histogram distribution of the image. As a result, the performance of the algorithm might be tested under several light conditions yielding a range of processing times in which the algorithm might perform. Table 3.3 shows the processing times for three cases of entropy. The timing elapsed in every step of the algorithm is presented in order to contrast the performance of the analog and digital processor. Recall that the decomposition and recomposition is performed by the digital processor, while most of the BEA processing is appointed to the analog one. The total processing time on each case highlight the entropic properties of our approach, in the sense that the processing time is greater as the dynamic range increases and the image tends to be uniform distributed. The equivalent frame rate is also presented to evaluate the capabilities of the approach to perform in real time. Note that such rates are good enough comparing to the implementation in Matlab. However, these figures can be improved significantly if the decomposing and recomposing timings were also improved, which represent together nearly half of the total processing time.

It is worth to remark that the NIOS processor is a 32-bit general purpose DSP at 200 MHz, which means that its processing power is not comparable to either common PC processors or the Q-Eye. As a result, the decomposition and recomposition represent a bottleneck for the algorithm. In fact, some data moving and control on the BEA are also performed by the NIOS. In order to overcome this problem, two solutions have been envisioned. First, Altera, the designer of NIOS II, has launched in past years a compiler called C2H, which accelerates the performance in 10 to 45 times the processing time by converting ANSI C functions into its corresponding VHDL description to be embedded in the FPGA. This solution suggests a great improvement in the total processing even in the most conservative case (i.e. 10x). On the other hand, the second solution would be changing the actual processor for another more powerful with image processing support. This is a logical next step already considered by Anafocus for the Eye-Ris vision system 2.0.

Table 3.3 Processing times of the BEA under different entropic conditions.

	Lap Timings				
Entropy	Decomposition	BEA	Recomposition	Total	Frame Rate
High	14 ms	64.86 ms	25 ms	103.86 ms	9.62 fps
Medium	14 ms	37.33 ms	25 ms	76.33 ms	13.1 fps
Low	14 ms	20.5 ms	25 ms	59.5 ms	16.8 fps

An important restriction of the binary erosion function provided by Q-Eye is that the structuring element cannot be defined by the user. At the moment, only a square or cross shaped structuring elements with a 3x3 neighborhood are defined. Therefore, if we want to emulate the effect of a bigger structuring element, we need to apply multiple erosions. Fortunately, this option is supported by the Q-Eye, which is performed in real time rather than in tandem, as commonly implemented in digital processors. Table 3.4 shows the processing times when multiple erosions are applied to an image with medium entropy. Note that the timings decrease as the number of erosions increases; this is an interesting result since normally in a digital processor the timing increases as the number of erosions does. This effect is caused by the fact that multiple erosions are performed at once, in real time, and given the processing of the upper bitplanes reduces the discriminative regions and thus the binary search in lower bitplanes.

From another point of view, we can explain this effect in the following way. Consider that the multiple erosions are performed in tandem; this means that the dynamic range of the image will decrease progressively moving the data to the lower part of the histogram. As a result, this fact is fully exploited by the BEA given its entropic properties.

Table 3.4 Timings for several numbers of erosions.

No. of Erosions	Timing	Frame Rate
1	76.33 ms	13.10 fps
3	73.00 ms	13.69 fps
5	65.46 ms	15.27 fps

Other basic morphological operators can be easily derived as a consequence of the BEA such as dilation, opening and closing. As discussed before, the dilation can be easily obtained by using the duality property, which yields the same processing times as the ones presented for the erosion. On the other hand, opening and closing operators can be derived by applying erosion and dilation operators in tandem. Unfortunately, under this scheme the BEA has to be applied two times, which increases the processing time. Table 3.5 shows the opening for different radius of a disk-shape structuring element.

Given that binary opening and closing are available in the Q-Eye, and thanks to the theoretical insight developed for the BEA, we have envisioned that it would be possible to develop an algorithm capable to perform gray-scale closing, opening and other rank filtering more efficiently. That is, each bitplane could be processed straightforward rather in tandem. This approach would yield a unified method based on bitwise processing improving the current timings. However, such approach requires further research since at the moment it is not possible to implement it in an iterative way to make it feasible to the Eye-RIS system.

Table 3.5 Opening timings for several neighborhood radiuses.

Radius of the SE	Timing	Frame Rate
1	124.87 ms	8.00 fps
2	125.03 ms	7.99 fps
3	123.2 ms	8.11 fps

Chapter 4

Conclusions

" When I examine myself and my methods of thought, I come to the conclusion that the gift of fantasy has meant more to me than any talent for abstract, positive thinking. "

-Albert Einstein-

4.1 Conclusions

A novel approach to compute basic morphological operators based on bitwise decomposition was presented in this work. Previous works related to the design and implementation of rank filters have been restricted by the conditions imposed by the threshold decomposition and stacking techniques. The major constrain says that the decomposition of a multilevel signal can only be manipulated with rank filters if the transformation that yields such decomposition is monotone increasing, which bitwise decomposition fails to fulfill. As a result, there exists very little literature related to morphological treatment of images under a bitwise manipulation given to this restriction.

The algorithm proposed exhibits diverse features and advantages that contrast to the approach based on threshold decomposition. For instance, the bitwise decomposition is a logarithmic representation of the gray-scale image, whereas the threshold decomposition depends linearly on the range of the image. Consequently, the number of bitplanes generated from the bitwise decomposition is considerable lower than threshold decomposition. This sole fact reduces significantly the complexity of the algorithm, since the decomposition and recomposition is performed in a natural fashion, whereas in TD bulky hardware and long time latencies are introduced. However, it should be also fair to mention that some overhead is introduced to transform the image through the binary search method, which consists on basic logic transformations.

Another important conclusion may be drawn from the analysis of complexity of both approaches. The number of bitplane erosions is the main time consuming process on each algorithm and the number of calls necessary to reach multilevel erosion is the same if the images are full in range. When an image is neither full in range nor uniform distributed, the bitwise erosion algorithm exploits its entropic properties reducing the number of bitplane erosions and thus the time processing.

The extension of the binary search method to images yields a very versatile algorithm, which not only can be used to derive erosion and dilation, but also it is able to generate a large spectrum of morphological transformations that lie in between at a minimum overhead cost. As a result, significant time processing and hardware can be reduced. We have also envisioned that the algorithm may be a strong prospect to be used in applications where the chip area is very constrained such as visual implants. This is achieved thanks to the extra time and hardware reduction obtained by applying linear quantization mapping, which generate some errors on the transformed image that are not visually perceptible.

4.2 Future Work

The bitwise erosion algorithm and its relation to the threshold decomposition has provided significant theoretical insight that goes beyond to the application of basic morphological operators such as erosion and dilation, since it can be also extended to other rank filters. The ability of the BEA to translate numeric addition to logic functions is a key tool that simplifies enormously the complexity of the algorithm. As a result, the list of properties that link the two approaches might be also helpful to deal with several levels of complexity in terms of chip integration. In other words, the algorithm can be implemented under different block structures in accordance with the designer's needs.

References

[1] R. Colin, "Vision chips mimic eye, brain functions", EE Times, June, 2005.

[2] G. Liñan, et al., "A 0.5˘m CMOS 10^6transistors analog programmable array processor for real–time image processing", Proc. of the 25th European Solid-State Circuits Conference, 21-23, pp. 358-361, 1999.

[3] J. Serra, "Introduction to mathematical morphology," Comput. Vision, Graphics, Image Processing, vol. 35, pp. 283-305, Sept. 1986.

[4] L. Abbott, R.M. Haralick and X. Zhuang, "Pipeline architectures for morphological image analysis," Machine Vision and Application, Vol.1, pp.23-40, 1988.

[5] K.I. Diamantaras, K.H. Zimmermann and S.Y. Kung, "Integrated fast implementation of mathematical morphology operations in image processing," *ZSCAS* 1990, pp.1442-1445.

[6] A.C.P. Loui, A.N. Venestsanopoulos, K.C. Smith and B.Benhabib, "Non-linear pipeline architecture for morphological signal processing," *ZSCAS* 1990, pp. 1438-1441.

[7] J. P. Fitch, E. J. Coyle, and N. C. Gallagher, "Median filtering by threshold decomposition," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-31, pp. 1183-1188, Dec. 1984.

[8] F. **Y.** Shih, O.R. Mitchell. "Threshold decomposition of gray-scale morphology into binary morphology" IEEE Trans. on Pattern Analysis and Machine Intelligence. 11, no. **1,** 31-42, 1989.

[9] C. L. Lee, J. S. Chen, C. H. Jen, "VLSI architecture for gray scale morphological filtering and its application", IEEE International Symposium on Circuit and Systems, 2100-2103 vol. 4, June, 1991.

[10] Ko, S. J., Morales A., Lee, K. H.: Fast recursive algorithms for morphological operators based on the basis matrix representation. IEEE Trans. IP 5, No. 6 (1996) 1073-1077

[11] P.Dudek and P.J.Hicks, "A General-Purpose Processor-per-Pixel Analog SIMD Vision Chip", IEEE Transactions on Circuits and Systems - I, vol. 52, no. 1, pp. 13-20, January 2005.

[12] Hardware Description of the Eye-RIS vision system, Anafocus, 2008.

[13] Henk J. A. M. Heijmans, "Mathematical Morphology: A Modern Approach in Image Processing Based on Algebra and Geometry", SIAM Rev. Volume 37, Issue 1, pp. 1-36, March 1995.

[14] S. R. Sternberg, "Grayscale morphology," IEEE Comput. Vision, Graphics, Image Processing, vol. *35,* pp. 333-355, Sept. 1986.

[15] J.P. Fitch, E.J. Coyle, and N.C. Gallaguer, "Median Filtering by Threshold Decomposition", IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-32, No. 6, Dec. 1984

[16] S.K. Mitra, M.R. Petraglia, and L. Onural, "Median Filtering by a Generalized Threshold decomposition", European Conference on Circuit Theory and Design, No 5-8, pp. 349-353, Sept. 1989.

[17] H. Zhang, and B. Yuan, "Generalized Threshold Decomposition", International Symposium on Speech, Image Processing and Neural Networks, pp. 49-52, April 1994, Hong Kong.

[18] D. Jabakhanji, and M. Shams, "A CMOS imager with on-pixel gray-scale erosion", 50th Midwest Symposium on Circuits and Systems, 2007. MWSCAS 2007, Volume , Issue 5-8, pp. 61 – 64 Aug. 2007.

[19] Datasheet of 8 bit digital comparator SN54ALS520, Texas Instruments, 1995.