Popular and Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

Colonel Ahmed Draia University of Adrar

Faculty of Sciences and Technology

Department of Mathematics and Computer Science

جامعة أحمد دراية.أدرار-الجزائر
Université Ahmed Draia.Adrar -Algérie

A Thesis Presented to Fulfill the Partial Requirement for Master’s Degree in Computer Science

Option: Network and Intelligent Systems

Title

Image Compression Using Logarithmic Functions with Pixels Shifting

Prepared by

Mekki BAROUDI

Jury members:

Dr. Mohammed OMARI	university of Adrar	supervisor
Mr. Mohamed Amine CHERAGUI	university of Adrar	president
Mr. Salah YAICHI	university of Adrar	examiner
Mr. Mohamed KOHILI	university of Adrar	examiner

Academic year 2015 / 2016

ABSTRACT

Digital image size or image stream size is too large and requires an amount of storage space and high bandwidth for transmission. Hence, image size in its original form needs to reduce in order to improve the storage space and accelerate the transmission. Therefore, image compression addresses the problem of reducing the amount of information required to represent a digital image. It is a process intended to yield a compact representation of an image. For this reason, many compression techniques have been proposed; some of these have been effective in some areas and failed in others. In this thesis we propose a new image compression technique called Logarithm Transform based image compression with pixels shifting. This technique has been proposed to achieve high compression ratios and high quality of the reconstructed image. In order to demonstrate the performance of the proposed method, a comparison between the proposed technique and other common compression techniques has been performed. It has been found that our method gives better compression ratios, especially for PSNR above 40(dB).

Keywords: lossy image compression, logarithm transform, pixels shifting, exponential transform, compression ratios, image quality.

ACKNOWLEDGMENTS

In preparing this thesis, I am highly indebted to pass my heartfelt thanks to many people who helped me in one way or another.

Above all, I would first like to thank my God.

I would like to thank my thesis advisor Dr. OMARI Mohammed of the Faculty of technology at University of ADRAR. The door to Prof. OMARI office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this thesis to be my own work, but steered me in the right direction whenever he thought I needed it.

My gratitude goes to all the faculty members of our mathematical and computer science department for their continuous help during the last five years. My thanks go to the defense jury: Mr. Mohamed Amine CHERAGUI, Mr. Mohamed KOHILI and Mr. Salah YAICHI for accepting to examine my modest work.

I would also like to acknowledge Mlle. MANSOURI Myra as the second reader of this work, and I am gratefully indebted to her for her very valuable comments on this thesis.

I am blessed with wonderful friends that I could not state one by one here in many ways, my successes are theirs, too. Most importantly, HACHEMI Mohamed Lotfi.

Finally, I must express my very profound gratitude to my parents and to my family for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

To my parents

Author

BAROUDI Mekki

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

"One picture is worth more than ten thousand words."
Anonymous.

Nowadays, the digital image is a substantial key in our life. This importance of digital image came with increasing development of technology in our world such as digital cameras, smartphones, numerical televisions, and personal computers, etc. Throughout the developmental history of digital image tools, we find several kinds of image such as binary image, gray-scale image, color image, and HD image. This progressive development is guided by two essentials requirements: high quality and high resolution of image.

Uncompressed multimedia (graphics, audio and video) data requires considerable storage capacity and transmission bandwidth. Despite rapid progress in mass-storage density, processor speeds, and digital communication system performance, demand for data storage capacity and data-transmission bandwidth continues to outstrip the capabilities of available technologies. The recent growth of data intensive multimedia-based web applications have not only sustained the need for more efficient ways to encode signals and images but have made compression of such signals central to storage and communication technology (Sethy & Swastik, 2009).

For this demand, image compression addresses the problem of reducing the amount of information required to represent a digital image. It is a process intended to yield a compact representation of an image, thereby reducing in file size which allows more images to be stored in a given amount of disk and more memory space. It also reduces the time required for image to be sent over the internet or downloaded from web pages. There are two major categories of compression algorithms: lossy and lossless. Unlike lossless kind, the lossy image compression allows to lose some information to achieve better compression ratio as much as possible, with acceptable degradation in quality of the reconstructed image.

Many people may have heard of JPEG (Joint Photographic Experts Group) and MPEG (Moving Pictures Experts Group), which are standards for representing images and video. Through the history of image compression, many approaches and schemes have been proposed in order to reach the goal of image compression. These methods have been satisfactory but they have not met all the demands of image compression.

In this thesis, we introduce a new image compression scheme: Logarithm Transform Based Image Compression with Pixels Shifting. As our proposed method is a lossy approach, it follows the general scheme of the lossy image compression. In the first stage, we propose a transformation based on the logarithmic numbers system (LNS) properties with pixels shifting; the next stage is the quantization, where we propose to use a non-uniform quantization by compression function; and in the last stage, we apply an efficient entropy encoding technique.

Our method has been proposed in order to provide sufficient high compression ratios with no appreciable degradation of image quality.

This thesis is organized into four chapters:

The First chapter presents some basics of theory of color and its concepts in section 1, and sets basic definitions of digital images, its types, characteristics, and its formats, in section 2.

The second chapter presents the general scheme of image compression, its benefits, characteristic to judge compression algorithm in section 1; and different types of image compression techniques in section 2.

The third chapter presents mathematically the logarithmic functions and its inverse which is the exponential functions with some properties and their domains application in section 1. In section 2, we present two related works: LOG-EXP based image compression and modified LOG-EXP based image compression and its evaluations.

The fourth chapter presents a detailed description of our proposed method in section 1; and in section 2, presents the implementation and the results obtained from each experiment based on the different parameters used.

Lastly, the thesis is closed with a general conclusion and some insights on future work.

CHAPTER 1
DIGITAL IMAGE and COLOR NOTIONS

INTRODUCTION

Nowadays, the digital image is a substantial key in our life. This interest of digital image came with increasing development of technology in our world such as digital cameras, smartphones, numerical televisions, and personal computers, etc. through the digital image tools development history, we find several kinds of image such as binary image, gray-scale image, color image, and HD image. This progressive development is guided by two essentials properties: high quality and high resolution of image. In this chapter we present a background of color concepts and notions in the first section, and in the second section we introduce the digital image and some related concepts.

SECTION 1: COLOR CONCEPTS AND BASICS

The human eye and brain together translate light into color. Light receptors within the eye transmit messages to the brain, which produces the familiar sensations of color (Curry, Matinsen, & Hopper, 2003).

1. The Concept of Light

1.1 What is light?

Light is a form of energy that has two aspects:

- An electromagnetic wave aspect
- A corpuscular aspect (photons)

The light emitted by the sun travels at a speed of approximately 300,000 km/s with a frequency of approximately 600,000 GHz.

1.2 The concept of color

The color of light is depends on its frequency, which itself depends on the wavelength and the speed of the wave front. The wavelength of an oscillatory phenomenon is usually characterized by the relation: $\lambda = CT$, where:

λ: represents the wavelength.

C: represents the speed of the wave fronts.

T: indicates the period of the wave (in seconds).

Radiation comprising only one wavelength is called monochromatic radiation and radiation which contains several wavelengths is called polychromatic radiation. The collection of all the wavelengths composing polychromatic radiation (and their respective luminous intensities) is called the spectrum.

However, the human eye is not capable of distinguishing the various components of this radiation and perceives only the result, which is a function of the different wavelengths which it comprises and their respective luminous intensity.

The human eye is able to see radiation with wavelengths between 380 and 780 nanometers. The radiation with wavelengths below 380nm is called ultraviolet radiation, while the radiation with wavelengths above 780nm tis known as infra-red radiation. The range of wavelengths that are visible to the human eye is called the "visible spectrum".

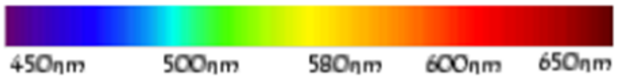


Figure 1: radiation space observed by human eye

It is possible to separate the spectral colors using a crystal prism.

1.3 The operation of the human eye

Thanks to the cornea (the translucent envelope of the eye) and the iris (which by closing allows the amount of light entering the eye to be regulated), an image is formed on the retina. The latter is made up of rods and cones.

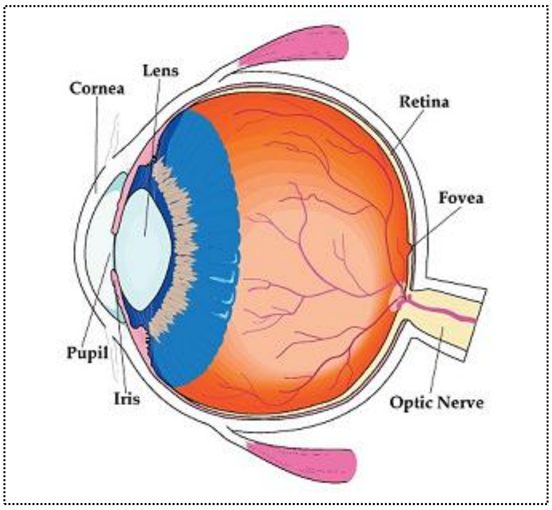


Figure 2: the human eye components

The rods, which contain a pigment called rhodopsin and are located in periphery of the retina, make it possible to perceive luminosity and movement (scotopic vision), while the cones, located in a region called the fovea, make it possible to differentiate the colors (photopic vision). There are actually three kinds of cones:

- Kind sensitive to red radiation (570 nm)
- Kind sensitive to green radiation (535 nm)
- Kind sensitive to blue radiation (445 nm)

Thus, when a type of cone is lacking, the perception of the colors is imperfect. This condition is known as daltonism (or dichromasy). Depending on the type of defective cone, people with this vision anomaly are known as:

- Protanopes, who are highly insensitive to red
- Deuteranopes who are highly insensitive to green
- Trinatopes who are highly insensitive to blue

In addition, it should be noted that the sensitivity of the human eye to luminous intensities related to the three primary colors is not the same:

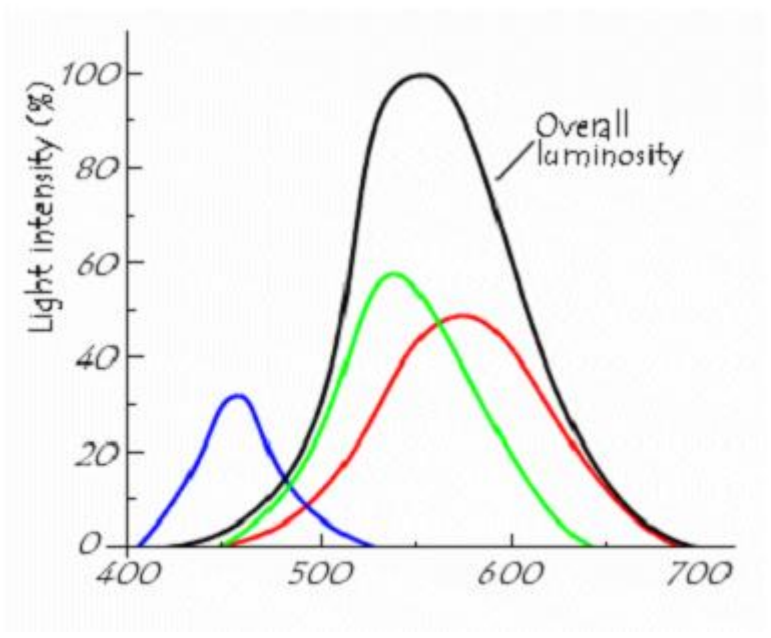


Figure 3: representation of human eye sensibility to the three primary colors

1.4 Additive and subtractive synthesis

There are two types of color synthesis:

Additive synthesis results from the addition of light components. The light components are directly added to the emission; this is the case of monitors or color televisions (Paquel, 1993). When the three components, red, green, blue (RGB) are added, white is obtained. The absence of components yields black. The secondary colors are cyan, magenta and yellow because:

- Green combined with blue yields cyan
- Blue combined with red yields magenta
- Green combined with red yields yellow

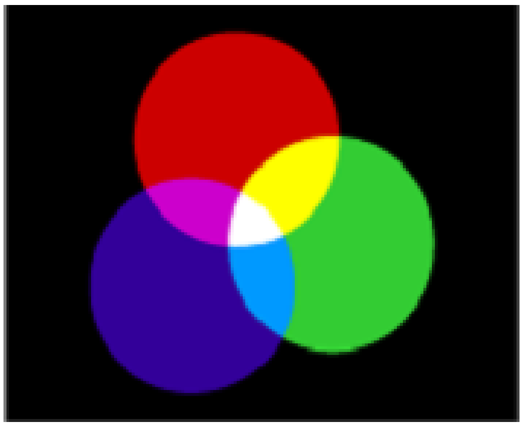


Figure 4: Additive principle of color combining (light)

Subtractive synthesis makes it possible to restore a color by subtraction, starting from a white light source, with filters for the complementary colors: yellow, magenta, and cyan. The addition of filters for all three colors yields black and their absence produces white.

When light falls on an object, some wavelengths are subtracted since they are absorbed by the object. What we see is the combination of the wavelengths that are reflected or transmitted (i.e. those that are not absorbed). This process is used in photography and for the impression of colors. The secondary colors are blue, the red and the green (Cavet, 1992):

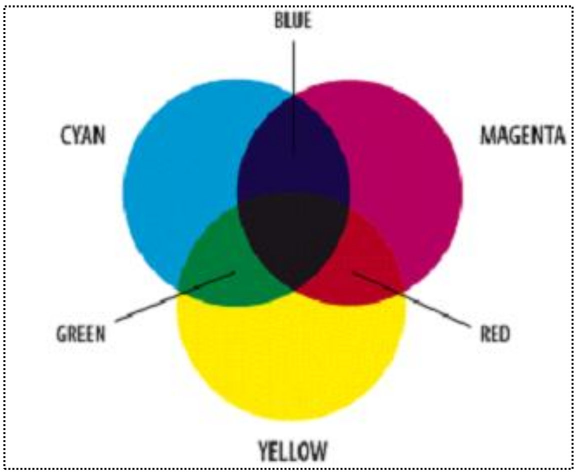


Figure 5: subtractive principle of color combining (light)

- Magenta combined with cyan yields blue
- Magenta combined with yellow yields red
- Cyan combined with yellow yields green

Two colors are said to be "complementary" if white is obtained by their additive synthesis, or black is obtained by their subtractive synthesis.

2. Computer Representation of Color

Color of an object can be considered as one of the physical properties of the object, along with, for example, its weight, size or shape. However, it differs from all other physical properties. To perceive or measure color of an object it must be exposed to light and unlike all of the other physical properties, color can be judged through no other sense than vision, while it is possible, for example, estimating the weight or the size by looking at the object or by touching it. To summarize, color of an object is a property which can be perceived by an observer through his vision only when there is an interaction with light (Boynton, 1996).

2.1 Color representations

In order to be able to handle colors correctly and to exchange colorimetric information it is necessary to have the means to categorize them and to choose them. Thus, it is not uncommon to have the choice of the color of a product before it is even manufactured. In this case, a palette is presented, in which the appropriate color is selected. Most of the time, the product (vehicle, building, etc.) has a color which corresponds to that selected.

Likewise in Information Technology, it is essential to have a means of choosing a color among all those that are usable. However, the possible color range is very vast and the image processing chain passes through various peripheral devices: for example, a digitizer (scanner) followed by image improvement software, and finally a printer. It is thus necessary to be able to represent the color in a reliable way in order to ensure coherence between these various peripheral devices (AMEY, 2001-2002).

The mathematical representation of a set of colors is thus called the "color space". There are several, among which the best known are:

- RGB Coding (Red, Green, Blue)
- HSL Coding (Hue, Saturation, Luminance)
- CMYK Coding
- CIE Coding
- YUV Coding
- YIQ Coding

The color spectrum that a peripheral display device allows to be shown is called a **gamut** or colorimetric space. The colors not belonging to the gamut are called out-of-range colors.

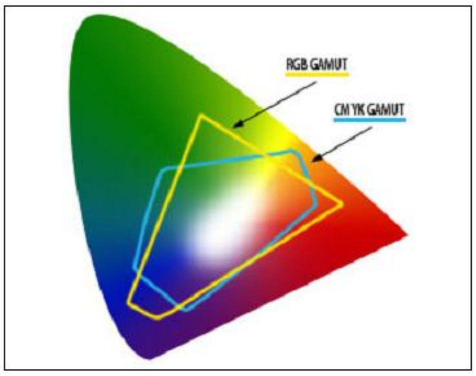


Figure 6: This illustration clearly shows the different gamut of the RGB and CMYK color spaces. The background is the CIE Chromaticity Diagram (representing the whole gamut of human color perception). From adobe.com.

2.2 RGB coding

The RGB coding (Red, Green, and Blue), developed in 1931 by the International Lighting Commission (Commission Internationale de l'Eclairage, CIE) consist in representing the color space with three monochromatic rays, with the following colors:

- Red (with a wavelength of 700.0 nm),

- Green (with a wavelength of 546.1 nm),
- Blue (with a wavelength of 435.8 nm)

This color space corresponds to the way in which the colors are usually coded on a computer, or more precisely to the way in which the computer screen cathode tubes represent the colors.

This color space corresponds to the way in which the colors are usually coded on a computer, or more precisely to the way in which the computer screen cathode tubes represent the colors. Thus, the RGB model proposes each color component to be coded on one byte, which corresponds to 256 intensities of red (2^8), 256 intensities of green and 256 intensities of blue, thus, there are 16777216 theoretical possibilities of different colors, that is, many more than the human eye can distinguish (approximately 2 million). However, this value is only theoretical because it strongly depends on the display device being used[1]. Since RGB coding is based on three components with the same proposed value range, it is usually graphically represented by a cube of which each axis corresponds to a primary color:

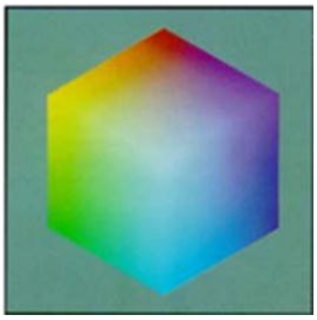


Figure 7: RGB space

2.3 CMYK (CMJN) Coding

CMY Coding (Cyan, Magenta, and Yellow) is to subtractive synthesis, what RGB coding is to additive synthesis. This model consists in breaking up a color into values of Cyan, Magenta and Yellow. The absence of these three components yields white while their sum yields black. However, since the black obtained by the sum of

[1] See (AMEY, 2001-2002)

the three colors Cyan, Magenta and Yellow is only partially black in practice (and is expensive), printers add a black ink component called pure black. This is known as the **quadrichromy**, or **CMYK model**. Just as the primary colors of CMY are the secondary colors of RGB, the primary colors of RGB are the secondary colors of CMY. But as the illustrations show, the colors created by the subtractive model of CMY don't look exactly like the colors created in the additive model of RGB. Particularly, CMY cannot reproduce the brightness of RGB colors (Few, February 2008).

2.4 CIE / Lab (L*a*b) coding

Colors may be perceived differently depending on individuals and may be displayed differently depending on the peripheral display devices. The International Lighting Commission (Commission Internationale de l'Eclairage, CIE) thus defined standards allowing a color to be defined, independently from the peripheral devices used. For this purpose, the CIE defined criteria based on the perception of color by the human eye, thanks to a triple stimulus[1].

In 1931, the CIE worked out the **xyY** colorimetric system which allows colors to be represented according to their chromaticity (X and Y axes) and their luminance (Y-axis). The chromaticity diagram (or chromatic diagram) resulting from a mathematical transformation represents pure colors on the periphery, i.e. the monochromatic radiation corresponding to the colors of the spectrum (the colors of the rainbow) indicated by their wavelength. The line connecting the ends of the diagram (thus connecting the two ends of the visible spectrum) is known as the line of purples, because it corresponds to the color purple, made up of the two monochromatic rays blue (420nm) and red (680 nm).

The gamut of a display device is generally represented by tracing, in the chromatic diagram, a polygon containing all the colors which it is capable of producing. However, this purely mathematical type of representation does not take into consideration physiological factors, i.e. color perception by the human eye,

[1] See (AMEY, 2001-2002)

which results in a chromaticity diagram that leaves too much space for greens (M.Irani, 1996).

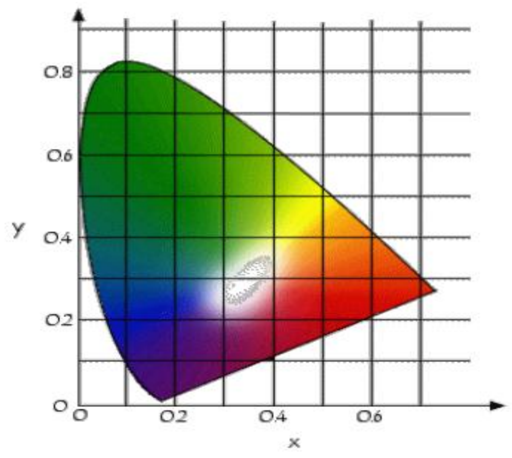


Figure 8: CIE system

In 1960 the **CIE** developed the **Lu*v*** model. Finally in 1976, in order to overcome the problems with the **xyY**, model, the **CIE** developed the **La*b*** colorimetric model (also known as the CIELab), in which a color is indicated by three values: **L**, luminance, expressed as a percentage (0 for black to 100 for white) **a** and **b** two color ranges, from green to red and from blue to yellow respectively, with values ranging from -120 to +120. The Lab mode thus covers the entire spectrum visible to the human eye and represents it in a uniform way. It thus makes it possible to describe all the visible colors in a way that is independent of any graphic technology. It thus includes all the **RGB** and **CMYK** colors; this is why software such as *Photoshop* uses this mode to go from one representation model to another[1].

It is a mode that is very much used in industry, but not much favored by most software since it is difficult to handle. The **CIE** models are not intuitive, however the use of these guarantees that a color created according to these models will be seen in the same way by everyone[2].

[1] See (M. Irani, 1996)
[2] See (Cavet, 1992)

2.5 YUV Coding

YUV was originally used for PAL[1] (European standard) analog video[2]. To convert from **RGB** to **YUV** spaces, the following equations can be used:

$$Y = 0.299\ R + 0.587\ G + 0.114\ B$$

$$U = 0.492\ (B - Y)$$

$$V = 0.877\ (R - Y)$$

Any errors in the resolution of the luminance (Y) are more important than the errors in the chrominance (U, V) values. The luminance information can be coded using higher bandwidth than the chrominance information.

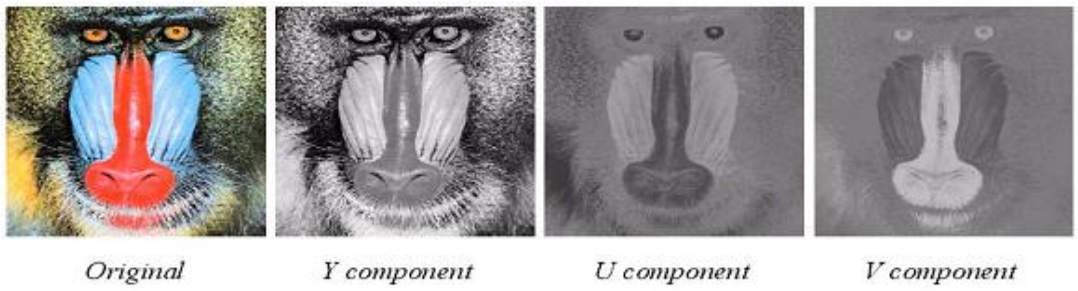


Figure 9: Example of YUV Space

2.6 HSL coding

The HSL Model (Hue, Saturation, Luminance), based on the work of painter Albert H.Munsell (who created the Munsell Atlas), is a representation model known as "natural", that is, close to the physiological perception of color by the human eye. Indeed, the RGB model, however well adapted it may be for the computer representation of colors or for showing them on peripheral display devices, does not allow colors to be selected easily. Indeed, RGB color adjustment with computer tools is generally done by using three slide blocks or of three cells with the relative values for each primary component, however the lightening of a color requires the respective values of each component to be increased proportionally. Thus, the HSL model was developed in order to overcome this issue with the RGB model.

[1] PAL (Phase Alternation Line)
[2] See (M. Irani, 1996)

The meaning of luminance and chrominance (Hue, and Saturation) is described as follows:

- **Luminance**: received brightness of the light, which is proportional to the total energy in the visible band.
- **Chrominance**: describe the perceived color tone of a light, which depends on the wavelength composition of light chrominance is in turn characterized by two attributes – hue and saturation.
  - **Hue**: Specify the color tone, which depends on the peak wavelength of the light.
  - **Saturation**: Describe how pure the color is, which depends on the spread or bandwidth of the light spectrum.

The following is a graphical representation of the HSL model, in which the color is represented by a chromatic circle and the luminance and saturation are represented by two axes:

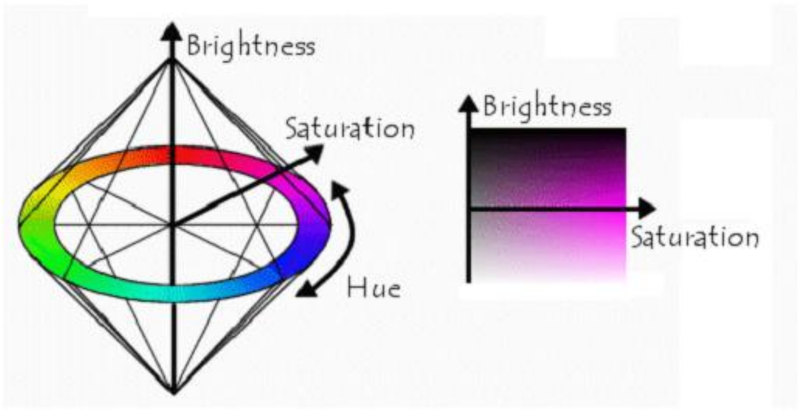


Figure 10: graphical representation of HSL coding

YCrCb is a subset of **YUV** that scales and shifts the chrominance values into the range of 0 to 1. The Y, **Cb**, and **Cr** components of one color image are defined in **YUV** color coordinate, where **Y** is commonly called the luminance and **Cb**, **Cr** are commonly called the chrominance.

The **RGB** primary commonly used for color display mixes the luminance and chrominance attributes of a light. In many applications, it is desirable to describe a color in terms of its luminance and chrominance content separately, to enable more efficient processing and transmission of color signals. Towards this goal, various three-component color coordinates have been developed, in which one component reflects the luminance and the other two collectively characterize hue and saturation. One such coordinate is the **YUV** color space. The [**Y Cb Cr**] values in the **YUV** coordinate are related to the [**R G B**] values in the **RGB** coordinate by

$$\begin{aligned} Y &= 0.299\,R + 0.587\,G + 0.114\,B \\ Cr &= ((B - Y) / 2) + 0.5 \qquad (1.1) \\ Cb &= ((R-Y)/1.6)+0.5 \end{aligned}$$

Similarly, if we would like to transform the YUV coordinate back to RGB coordinate, the inverse matrix can be calculated from (1.1), and the inverse transform is taken to obtain the corresponding RGB components (Wei, p. 02).

SECTION 2: DIGITAL IMAGE

1. Digital Image Definitions

1.1 Definition 1

A digital image is represented by a two-dimensional array of pixels, which are arranged in rows and columns. Hence, a digital image can be presented as M×N array (Rfael & Richard, 2002).

$$(x, y) = \begin{bmatrix} f(0,0) & \cdots & f(0, N-1) \\ \vdots & \ddots & \vdots \\ f(M-1, 0) & \cdots & f(N-1, M-1) \end{bmatrix}$$

Where *f (0, 0)* gives the pixel of the left top corner of the array that represents the image and *f (M-1, N-1)* represents the right bottom corner of the array. A Grey-scale image, also referred to as a monochrome image contains the values ranging from 0 to 255, where 0 is black, 255 is white and values in between are shades of grey. In color images, each pixel of the array is constructed by combining three different channels (RGB) which are R=red, G=green and B=blue. Each channel represents a value from 0 to 255. In digital image, each pixel is stored in three bytes, while in a Grey image is represented by only one byte. Therefore, color images take three times the size of Gray images (Firas A. Jassim, October 2012).

1.2 Definition 2

A digital image *a [m, n]* described in 2D discrete space is derived from an analog image *a(x, y)* in a 2D continuous space through a *sampling* process that is frequently referred to as digitization. For now we will look at some basic definitions associated with the digital image. The effect of digitization is shown in Figure 12.

The 2D continuous image *a(x, y)* is divided into *N rows* and *M columns*. The intersection of a row and a column is termed a pixel. The value assigned to the integer coordinates *[m, n]* with {m=0,1,2,…,M–1} and {n=0,1,2,…,N–1} is *a[m, n]*. In fact, in most cases *a(x, y)* – which we might consider to be the physical signal that impinges on the face of a 2D sensor – is actually a function of many variables including depth (z), color (λ), and time (t). Unless otherwise stated, we will consider the case of 2D, monochromatic, static images in this chapter.

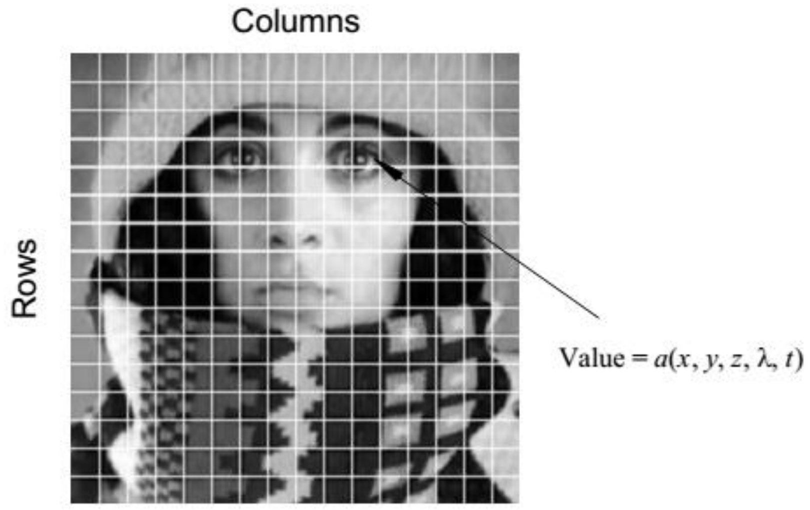


Figure 11: Digitization of a continuous image. The pixel at coordinates [m=10, n=3] has the integer brightness value 110.

The image shown in Figure 12 has been divided into *N* = 16 rows and *M* = 16 columns. The value assigned to every pixel is the average brightness in the *pixel* rounded to the nearest integer value. The process of representing the amplitude of the 2D signal at a given coordinate as an integer value with L different gray levels is usually referred to as amplitude quantization or simply *quantization* (Ian , Jan , & Lucas , 2007).

1.3 Common values

There are standard values for the various parameters encountered in digital image processing. These values can be caused by video standards, by algorithmic requirements, or by the desire to keep digital circuitry simple. Table 1 gives some commonly encountered values.

Table 1: Common values of digital image parameters

Parameter	Symbol	Typical values					
Rows	N	256	512	525	625	1024	1080
Columns	M	256	512	768	1024	1920	-
Gray levels	L	2	64	256	1024	4096	16384

Quite frequently we see cases of $M=N=2K$ where $\{K = 8, 9, 10, 11, 12\}$. This can be motivated by digital circuitry or by the use of certain algorithms such as the (fast) Fourier transform. The number of distinct gray levels is usually a power of 2, that is, $L=2B$ where B is the number of bits in the binary representation of the brightness levels. When $B>1$ we speak of a *gray-level image*; when $B=1$ we speak of a *binary image*. In a binary image there are just two gray levels which can be referred to, for example, as “black” and “white” or “0” and “1”[1].

2. Computer Graphics

The field of the IT concerning the creation and the handling of digital images is called computer graphics. Computer graphics cover various areas of knowledge,

[1] See (Ian , Jan , & Lucas , 2007).

including the representation of graphic elements (text, image or video), as well as their transformations (rotation, translation, zoom, etc.) by means of algorithms (Salles, mars 2005).

2.1 Display technology

The image is shown on a screen (also called a monitor), which is an output peripheral device that allows a visual representation to be offered. This information comes from the computer, but in an "indirect" way. Indeed, the processor does not directly send information to the monitor, but processes the information coming from its Random access memory (RAM), then sends it to a graphics card that converts the information into electrical impulses, which it then sends to the monitor.

Computer monitors are usually cathode tubes, i.e. a tube made out of glass in which an electron gun emits electrons which are then directed by a magnetic field towards a screen on which small phosphorescent elements (luminophores) are laid out, constituting points (pixels) that emit light when the electrons hit them[1].

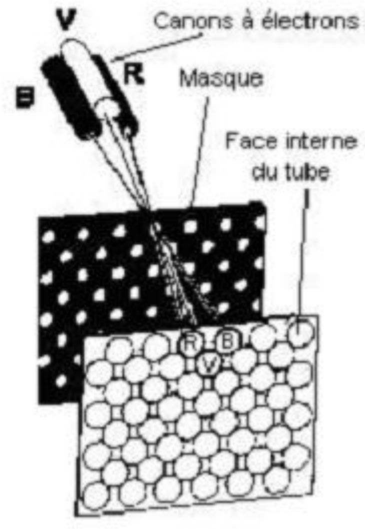


Figure 12: Computer monitors principle

An image consists of a set of points called pixels (the word pixel is an abbreviation of PICture ELement) the pixel is thus the smallest component of a

[1] See (M. Irani, 1996)

digital image. The entire set of these pixels is contained in a two-dimensional table constituting the image:

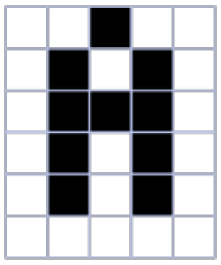


Figure 13: The pixel: is thus the smallest component of a digital image

Since screen-sweeping is carried out from left to right and from top to bottom, it is usual to indicate the pixel located at the top left hand corner of the image using the coordinates [0,0], this means that the directions of the image axes are the following:

- The direction of the X-axis is from left to right
- The direction of the Y-axis is from top to bottom, contrary to the conventional notation in mathematics, where the direction of the Y-axis is upwards.

2.2 Definition and resolution

The number of points (pixels) constituting the image, that is, its “dimensions” (the number of columns of the image multiplied by its number of rows) is known as the **definition**. An image 640 pixels wide and 480 pixels high will have a definition of 640 by 480 pixels, which is written as 640x480.

On the other hand, the **resolution**, a term often confused with the “definition”, is determined by the number of points per unit of area, expressed in dots per inch (**DPI**), an inch being equivalent to 2.54 cm. The resolution thus makes it possible to establish the relationship between the number of pixels of an image and the actual size of its representation on a physical support. A resolution of 300 dpi thus means 300 columns and 300 lines of pixels in a square inch which thus yields 90000 pixels in a square inch. The 72 dpi reference resolution gives us a

1"/72 (an inch divided by 72) pixel, that is to say 0.353mm, corresponding to a pica (Anglo-Saxon typographical unit)[1].

2.3 Color models

An image is thus represented by a two-dimensional table in which each cell is a pixel. To represent an image by means of computer, it is thus enough to create a pixel table in which each cell contains a value. The value stored in a cell is coded on a certain number of bits which determine the color or the intensity of the pixel; this is called the **coding depth** (or is sometimes also called the **color depth**). There are several coding depth standards:

- **Black** and **white bitmap**: by storing one bit in each cell, it is possible to define two colors (black or white).
- **Bitmap** with **16 colors** or **16 levels of grey**: storing 4 bits in each cell, it is possible to define 24 intensities for each pixel, that is, 1 6 degrees of grey ranging from black to white or 16 different colors.
- **Bitmap** with **256 colors** or **256 levels of grey**: by storing a byte in each cell, it is possible to define 24 intensities, that is, 256 degrees of grey ranging from black to white or 256 different colors.
- **Color palette** (**color map**): thanks to this method it is possible to define a pallet, or color table, with all the colors that can be contained in the image, for each of which there is an associated index. The number of bits reserved for the coding of each index of the palette determines the number of colors which can be used. Thus, by coding the indexes on 8 bits it is possible to define 256 usable colors; that is, each cell of the two-dimensional table that represents the image will contain a number indicating the index of the color to be used. An image whose colors are coded according to this technique is thus called an indexed color image (Rabbani & Jones, 1991).

[1] See (M. Irani, 1996)

- "**True Colors**" or "**real colors**": this representation allows an image to be represented by defining each component (RGB, for red, green and blue). Each pixel is represented by a set comprising the three components, each one coded on a byte, that is, on the whole 24 bits (16 million colors). It is possible to add a fourth component, making it possible to add information regarding transparency or texture; each pixel is then coded on 32 bits.

2.4 Weight of an image

To know the weight (in bytes) of an image, it is necessary to count the number of pixels that the image contains, which amounts to calculating the number of cells in the table , that is to say, the height of the table multiplied by its width. The weight of the image is thus equal to its number of pixels multiplied by the weight of each one of these elements.

The following is the calculation for a 640x480 True Color image:

Number of pixels: 640 x 480 = 307200 with 3 bytes per pixel

The weight of the image is thus equal to: 307200 x 3 = 921600 bytes=900KB

To know the size in KB, it suffices to divide by 1024. The following are some examples (considering that the image is not compressed):

Table 2: Some examples of weights of diffrents image color depth

Image definition	Black and white(1 bit)	256 colors (8 bits)	65000 colors (16 bits)	True color (24 bits)
320x200	7.8 KB	62.5 KB	125 KB	187.5 KB
640x480	37.5 KB	300 KB	600 KB	900 KB
800x600	58.6 KB	468.7 KB	937.5 KB	1.4 MB
1024x768	96 KB	768 KB	1.5 MB	2.3 MB

This shows the amount of video memory that your graphics card needs depending on the screen definition (number of points displayed) and on the number of colors. The example thus shows that a chart, having at least 4 MB of video memory, is needed in order to show a resolution of 1024x768 in true color…

3. Images Coding

3.1 Vector images and bitmap images

There are two main categories of images (THON, 2013-2014):

- **Bitmap images** (also called raster images): these are pixeled images, i.e. a set of points (pixels) contained in a table, each of these points having one or more values describing its color.
- **Vector images**: vector images are representations of geometrical entities such as circles, rectangles or segments. These are represented by mathematical formulae (a rectangle is defined by two points, a circle by a center and a radius, a curve by several points and an equation). The processor will "translate" these forms into information that a can be interpreted by the graphics card.

Since a vector image is only made up of mathematical entities, it is possible to easily apply geometrical transformations to it (zoom, stretching…), while a bitmap image, made up of pixels, will not be able to undergo such transformations without suffering a loss of information called **distortion**. The appearance of pixels in an image following a geometrical transformation (in particular when enlarging it) is called **pixellation** (also known as aliasing). Moreover, vector images (called clipart in the case of a vector object) make it possible to define an image with very little information, which makes the files quite small.

On the other hand, a vector image only allows simple forms to be represented. Although it is true that a superposition of various simple elements can yield very impressive results, it isn't possible to describe all images by vectors; this is particularly the case of realistic photographs.

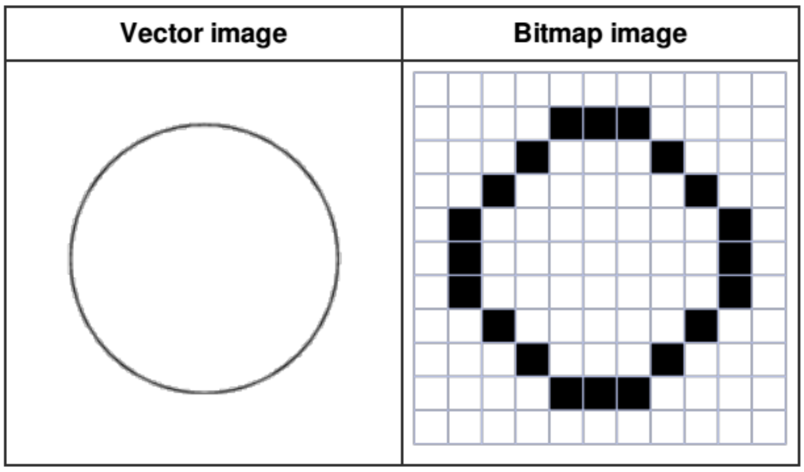


Figure 14: Comparison between vector and bitmap images

The “vector” image above is just a representation of what a vector image could resemble, because the quality of the image depends on the device used to make it visible to the eye. Your screen probably allows you to see this image with a resolution of at least 72 pixels per inch; the same file printed on a printer would offer a better image quality because it would be printed using at least 300 pixels per inch. Thanks to the technology developed by **Macromedia** and its software **Macromedia Flash**, or the **SVG** plug-in the vector format can be used on the Internet today.

4. Graphic File Formats

4.1 What is a file format?

We previously presented the way in which an image was coded to be shown on a monitor, however, when we want to store an image in a file, this format is not the most practical…

Indeed, we may want an image which takes up less memory space, or an image which can be increased in size without causing pixelation. Thus, it is possible to store the image, describing it with an equation, in a file with a data structure, which will need to be decoded by the processor before the information is sent to the graphics card:

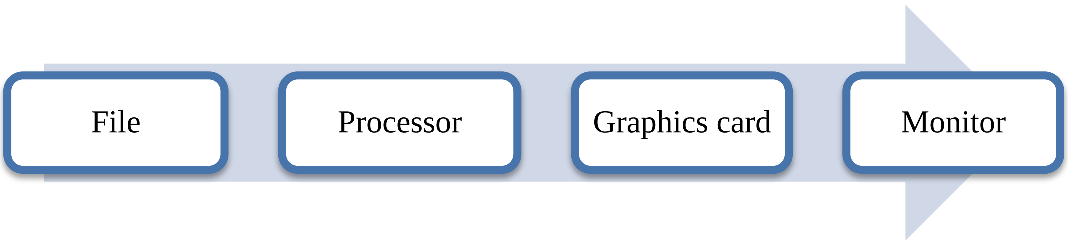


Figure 15: Process of computer information processing

4.2 Types of file formats

There are a great number of file formats. The most commonly used graphic file formats are the following[1]:

Table 3: Some famous file formats

Format	Compression	Maximum dimensions	Maximum number of colors
BMP	None /RLE	65536 x 65536	16777216
GIF	LZW	65536 x 65536	256
IFF	None /RLE	65536 x 65536	over 16,777,216
JPEG	JPEG	65536 x 65536	over 16,777,216
PCX	None /RLE	65536 x 65536	16 ,777,216
PNG	RLE	65536 x 65536	over 16,777,216
TGA	None /RLE	65536 x 65536	over 16,777,216
TIFF/TIF	Packbits / CCITT G3&4 / RLE / JPEG /LZW / UIT-T	2^{32} -1	over 16,777,216

5. The BMP Format

In this paragraph we shall consider the bitmap (.**BMP**) binary file and it structure to provide a foundation for the understanding of pixel color representation and the drawing of the information. We shall not consider every variation (and there are many) of these file but only this. The file formats are an essential precursor to the study of image processing methods and this knowledge will also support the work on texturing.

[1] See (THON, 2013-2014)

The **BMP** is (Whitrow, 2008) one of the simplest formats. It was jointly developed by Microsoft and IBM, which explains why it is particularly widespread on Windows and OS/2 platforms. Bitmap files are stored with the name extension .**BMP** and occasionally we also see bitmap files stored in a device-independent bitmap form with the name extension .**DIB**. Device-independent bitmap files are simply a list of pixels with values for the red, green and blue components and omit the header information associated with size and other descriptors.

The structure of a bitmap file is the following[1]:

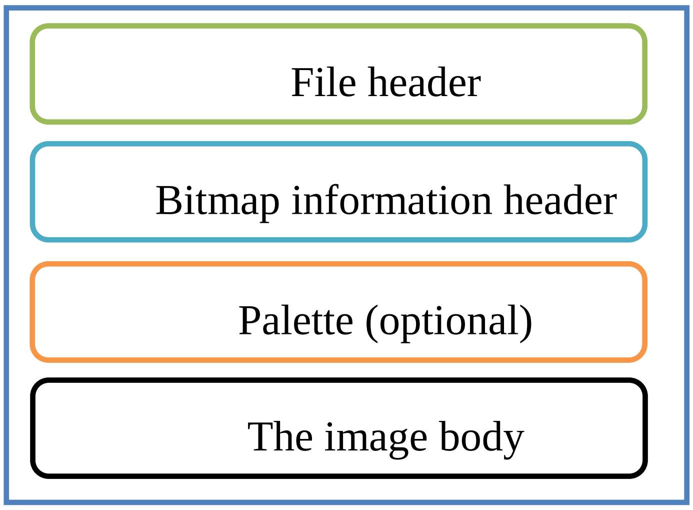


Figure 16: Structure of a bitmap file

5.1 File header

The file header provides information on the type of file (Bitmap) and its size as well as indicating where information concerning the image actually starts.

The header comprises four fields:

- The signature (on 2 bytes), indicating that it is a BMP file, using two characters
  - BM, 424D in hexadecimal, indicates that it is a Windows Bitmap.

[1] See also (THON, 2013-2014)

  - BA indicates that it is an OS/2 Bitmap.
  - CI indicates that it is an OS/2 color icon.
  - CP indicates that it is an OS/2 color pointer.
  - IC indicates that it an OS/2 icon.
  - PT indicates that it is an OS/2 pointer.
- The total size of the file in bytes (coded on 4 bytes)
- A reserved field (on 4 bytes)
- The offset of the image (on 4 bytes), that is, it is the location of the start of the image information relative to the start of the file.

5.2 Bitmap information header

The bitmap information header provides information on the image, particularly its dimensions and its colors.

The bitmap information header comprises four fields:

- The size of the bitmap information header in bytes (coded on 4 bytes). The following hexadecimal values are possible according to the type of BMP format:
  - 28 for Windows 3.1 x, 95, NT…
  - 0C for OS/2 1 .x
  - F0 for OS/2 2.x
- The width of the image (on 4 bytes), i.e. the number of pixels counted horizontally
- The height of the image (on 4 bytes), i.e. the number of pixels counted vertically
- The number of planes (on 2 bytes). This value is always 1
- The color model depth of (on 2 bytes), i.e. the number of bits used to code the color. This value may be equal to 1, 4, 8, 16, 24 or 32
- The compression method (on 4 bytes). This value is 0 when the image is not compressed, or either 1, 2 or 3 depending on the type of compression used:

    - 1 for RLE coding of 8 bits per pixel
    - 2 for RLE coding of 4 bits per pixel
    - 3 for bit field coding, meaning that the color is coded by a triple mask represented by the palette
- The total size of the image in bytes (on 4 bytes).
- The horizontal resolution (on 4 bytes), i.e. the number of pixels per meter counted horizontally
- The vertical resolution (on 4 bytes), i.e. the number of pixels per meter counted vertically
- The number of palette colors (on 4 bytes)
- The number of important palette colors (on 4 bytes). This field may be equal to 0 when all colors are important.

5.3 Image palette

The palette is optional. When a palette is defined, it contains 4 bytes successively for each of its entries, representing:

- The blue component (on one byte)
- The green component (on one byte)
- The red component (on one byte)
- A reserved field (on one byte)

5.4 Image coding

The coding of the image is done by successively writing the bits corresponding to each pixel, line by line, starting with the pixel in the bottom left-hand corner.

- 2-Color images use 1 bit per pixel, which means that one byte allows 8 pixels to be coded
- 16-Color images use 4 bits per pixel, which means that one byte allows 2 pixels to be coded

- 256-Color images use 8 bits per pixel, which means that one byte is needed to code each pixel
- Real color images use 24 bits per pixel, which means that 3 bytes are needed to code each pixel, taking care to respect the alternating color order for blue, green and red.

Each image line must comprise a total number of bytes that is a multiple of 4; if this is not the case, the line must be completed with as many 0 as necessary in order to respect this criterion.

*End of **BMP** specification.*

6. GIF Format

The **GIF** format (Graphic Interchange Format) is a bitmap (raster) graphic file format developed by CompuServe (Ohio, 1990).

There are two versions of this file format developed respectively in 1987 and 1989:

- **GIF 87a** supporting LZW compression, interlacing (allowing progressive display), a 256 color palette and the possibility animated images (called animated GIFs) by storing several images within the same file[1].
- **GIF 89a** adding the possibility of defining a transparent color to the palette and of specifying the time for animations[2].

6.1 Characteristics of the GIF format

A GIF image may contain between 2 and 256 colors (2, 4, 8, 16, 32, 64, 128 or 256) among 16.8 million in its palette. Thus, due to this palette with a limited number of colors (not limited in different colors), the images obtained by this format are generally very small in size.

[1] For more infos see: ftp://ftp.ncsa.uiuc.edu/misc/file.formats/graphics.formats/gif89a.doc
[2] For more infos see: ftp://ftp.ncsa.uiuc.edu/misc/file.formats/graphics.formats/gif87a.doc

However, given the proprietary character of the LZW compression algorithm, all software publishers handling GIF images must pay a royalty to Unisys, the company that is the holder of the rights. This is one of the reasons why the PNG format is becoming increasingly more popular, to the detriment of the GIF format.

*End of **GIF** specification.*

7. PNG Format

The PNG format (Portable Network Graphics or Ping format) is a (Adler.et.al., 1996) bitmap (raster) graphic file format. It was developed in 1995 in order to provide a free alternative to the GIF format, which is a proprietary format whose rights are held by Unisys (proprietor of the LZW compression algorithm), to whom all software publishers using this type of format are under obligation to pay royalties. Thus, PNG is also a recursive acronym for PNG Not GIF.

7.1 Characteristics of the PNG format

The PNG format makes it possible to store images in black and white (a color depth of 16 bits per pixel), true color (a color depth of 48 bits per pixel), as well as indexed images, using a palette of 256 colors.

Moreover, it supports alpha channel transparency, i.e. the possibility of defining 256 levels of transparency, while the GIF format only allows one color of the pallet to be defined as transparent. It also has an interlacing function which makes it possible to show the image gradually.

The compression offered by this format is a (lossless compression) 5 to 25% better than GIF compression.

Finally, PNG stores image gamma information, which makes a gamma correction possible and allows it to become display-device-independent. Error correction mechanisms are also stored in the file in order to guarantee its integrity.

7.2 Structure of a PNG file

A PNG file comprises a signature, making it possible to indicate that it is a PNG file, followed by a series of elements called chunks. The signature of a PNG file (in decimal notation) is the following:

1 | 137 80 78 71 13 10 26 10

1 | 89 50 4E 47 0D 0A 1 A 0A

Each chunk comprises 4 parts:

- The size, a non-signed 4-byte integer, describing the size of the chunk
- The chunk type: a 4-character (4-bytes) code comprised by ASCII alphanumeric characters (A-Z, a-z, 65 to 90 and 97 to 122) allowing the nature of the chunk to be established
- The chunk data
- The CRC (cyclic redundancy check), a 4-byte correcting code which allows the integrity of the chunk to be checked

The chunks can be present in any order except that they must start with the header chunk (IHDR chunk) and finish with the end chunk (IEND chunk)

The main chunks (called critical chunks) are:

- IHDR bitmap information header
- PLTE Palette
- IDAT Image data
- IEND Image trailer

The other chunks (called ancillary chunks) are the following:

- **bKGD** Background color
- **cHRM** Primary chromaticities and white point
- **gAMA** Image gamma
- **hIST** Image histogram
- **pHYs** Physical pixel dimensions

- **sBIT** Significant bits
- **tEXt** Textual data
- **tIME** Image last-modification time
- **tRNS** Transparency
- **zTXt** Compressed textual data

*End of **PNG** specification.*

CONCLUSION

In this first chapter, we have presented some basics of theory of color and its concepts in section 1; and digital image in section 2. As summary: An image consists of a set of points called pixels (the word pixel is an abbreviation of PICture ELement) the pixel is thus the smallest component of a digital image.

An image is thus represented by a two-dimensional table in which each cell is a pixel. To represent an image by means of computer, it is thus enough to create a pixel table in which each cell contains a value. The value stored in a cell is coded on a certain number of bits which determine the color or the intensity of the pixel. This is called the coding depth (or is sometimes also called the color depth). There are several coding depth standards: Black and white bitmap, Bitmap with 16 colors or 16 levels of grey, Bitmap with 256 colors or 256 levels of grey, Color palette (color map), and "True Colors" or "real colors".

Generally, the image size or image stream size is too large and requires amount of storage space or high bandwidth for communication, image size in its original form needs to reduce for improvement the storage space and accelerate transmission, this process is called image compression. In the next chapter we'll present the image compression basics concepts and techniques.

CHAPTER 2
IMAGE COMPRESSION BASICS AND TECHNIQUES

INTRODUCTION

Uncompressed multimedia (graphics, audio and video) data requires considerable storage capacity and transmission bandwidth. Despite rapid progress in mass-storage density, processor speeds, and digital communication system performance, demand for data storage capacity and data-transmission bandwidth continues to outstrip the capabilities of available technologies. The recent growth of data intensive multimedia-based web applications have not only sustained the need for more efficient ways to encode signals and images but have made compression of such signals central to storage and communication technology (Sethy & Swastik, 2009).

Image compression addresses the problem of reducing the amount of information required to represent a digital image. It is a process intended to yield a compact representation of an image, thereby reducing the image storage transmission requirements. There are two major categories of compression algorithms: lossy and lossless. Lossless compression reduces the number of bits required to represent an image such that the reconstructed image is numerically identical to the original one on a pixel-by-pixel basis. On the other hand, lossy compression schemes allow degradations in the reconstructed image in exchange for a reduced bit rate. These degradations may or may not be visually apparent, and greater compression can be achieved by allowing more degradation (Kurt & Fayez, June 1996).

SECTION 1: IMAGE COMPRESSION BASICS AND CONCEPTS

1. History of Data Compression

Data compression has only played a significant role in computing since the 1970s, when the Internet was becoming more popular and the Lempel-Ziv algorithms were invented, but it has a much longer history outside of computing. Morse code, invented in 1838, is the earliest instance of data compression in that the most common letters in the English language such as “e” and “t” are given shorter Morse codes. Later, as mainframe computers were starting to take hold in 1949, Claude Shannon and Robert Fano invented Shannon-Fano coding. Their algorithm assigns codes to symbols in a given block of data based on the probability of the symbol occurring. The probability is of a symbol occurring is inversely proportional to the length of the code, resulting in a shorter way to represent the data (Wolfram, 2002).

Two years later, David Huffman was studying information theory and had a class with Robert Fano. Fano gave the class the choice of writing a term paper or taking a final exam. Huffman chose the term paper, which was to be on finding the most efficient method of binary coding. After working for months and failing to come up with anything, Huffman was about to throw away all his work and start studying for the final exam in lieu of the paper. It was at that point that he had an epiphany, figuring out a very similar yet more efficient technique to Shannon-Fano coding. The key difference between Shannon-Fano coding and Huffman coding is that in the former the probability tree is built bottom-up, creating a suboptimal result, and in the latter it is built top-down (Huffman, September 1991).

Early implementations of Shannon-Fano and Huffman coding were done using hardware and hardcoded codes. It was not until the 1970s and the advent ofthe Internet and online storage that software compression was implemented that Huffman codes were dynamically generated based on the input data[1]. Later, in

[1] See (Wolfram, 2002).

1977, Abraham Lempel and Jacob Ziv published their groundbreaking LZ77 algorithm, the first algorithm to use a dictionary to compress data. More specifically, LZ77 used a dynamic dictionary oftentimes called a sliding window (Ziv & Lempel, 1977). In 1978, the same duo published their LZ78 algorithm which also uses a dictionary; unlike LZ77, this algorithm parses the input data and generates a static dictionary rather than generating it dynamically (Ziv & Lempel, 1978).

2. Why Compress Data?

Nowadays, the computing power of processors increases more quickly than storage capacities, and is much faster than network band-widths, because this requires enormous changes in the telecommunication infrastructures.

Thus, to compensate for this, it is usual to rather reduce the size of the data by exploiting the computing power of the processors rather than by increasing storage and data transmission capacities (Cohen & Resnikoff, 1994).

3. What Is Image Compression?

Image compression means minimizing the size in bytes of a graphics file without degrading the quality of the image to an unacceptable level. The reduction in file size allows more images to be stored in a given amount of disk more memory space. It also reduces the time required for image to be sent over the internet or downloaded from web pages.

In other word, Compression consists in reducing the physical size of information blocks. A compressor uses an algorithm which is used to optimize the data by using suitable considerations for the type as data to be compressed; a decompressor is thus necessary to reconstruct the original data using an algorithm that is the opposite to that used for compression (PLUME, 1993).

The compression method depends intrinsically on the type of data to be compressed: an image will not be compressed in the same way as an audio file… (WAGNER, 1993).

4. Need for Compression

An image 1024 pixel *1024 pixel *1024 pixel *24 bit , without compression would require 3 MB of storage and 7 minute of transmission, utilizing a high speed, 64 kbs, ISDN line . If the image is compressed at a 10:1 compression ratio, the storage requirement is reduced to 300 KB and the transmission time is reduced to less than 7 second. Images file in an uncompressed form are very large, and the internet especially for people using a 56 kbps dialup modem, can be pretty slow. This combination could seriously limit one of the webs most appreciated aspects – its ability to present images easily.

The Table below shows the qualitative transition from simple text to full-motion video data and the disk space, transmission bandwidth, and transmission time needed to store and transmit such uncompressed data.

Table 4: Multimedia data types and uncompressed storage space, transmission bandwidth, and transmission time required.

Multimedia	Size/ Duration	Bits/Pixel	Un-compressed Size	Transmission	Time
A page of text	11" x 8.5"	Varying Resolution	4-8 KB	32-64 Kb/page	1.1-2.2 sec
Telephone quality speech	10sec	8 bps	80 KB	64 Kb/sec	22.2 sec
Gray-scale Image	512 x 512	8 bpp	262 KB	2.1 Mb/image	1 min 13 sec
Color Image	512x512	24 bpp	786 KB	6.29 Mb/image	3 min 39 sec
Medical Image	2048 x 1680	12 bpp	5.16 MB	41.3 Mb/image	23 min 54 sec
SHD Image	2048x2048	24 bpp	12.58 MB	100Mb/image	58 m 15 s

Full-motion Video	640x480, 1 min (30 frames/sec)	24 bpp	1.66 GB	221 Mb/sec	5 days 8 hrs

The examples above clearly illustrate the need for sufficient storage space, large transmission bandwidth, and long transmission time for image. Audio and video data at present, the only solution is to compress multimedia data before its storage and transmission and decompress it at the receiver for play back for example with a compression ratio of 32:1, the space , bandwidth and the transmission time requirement can be reduced by a factor of 32 , with acceptable quality (Salomon, 1998).

5. Principles of Image Compression

Image compression (Sonal, 2015) addresses the problem of reducing the amount of data required to represent a digital image. It is a process intended to yield a compact representation of an image, thereby reducing the image storage/transmission requirements. Compression is achieved by the removal of one or more of the three basic data redundancies:

- Coding Redundancy
- Interpixel Redundancy
- Psychovisual Redundancy

Coding redundancy is present when less than optimal code words are used. Interpixel redundancy results from correlations between the pixels of an image. Psychovisual redundancy is due to data that is ignored by the human visual system (i.e. visually non-essential information).

Image compression techniques reduce the number of bits required to represent an image by taking advantage of these redundancies. An inverse process called decompression (decoding) is applied to the compressed data to get the reconstructed image. The objective of compression is to reduce the number of bits as much as possible, while keeping the resolution and the visual quality of the reconstructed image as close to the original image as possible. Image compression systems are composed of two distinct structural blocks: an encoder and a decoder.

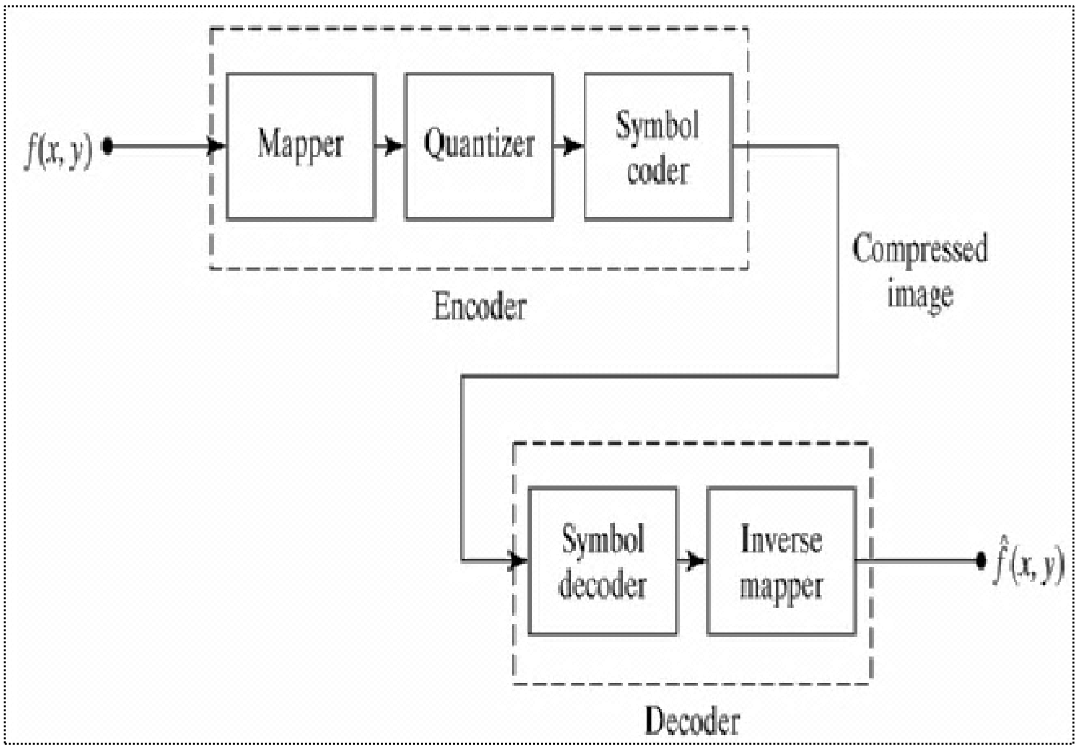


Figure 17: Image compression system (Mozammel & Amina, 2012)

Image f(x,y) is fed into the encoder, which creates a set of symbols form the input data and uses them to represent the image. If we let n1 and n2 denote the number of information carrying units(usually bits) in the original and encoded images respectively, the compression that is achieved can be quantified numerically via the compression ratio,   CR = n1 /n2.

As shown in the figure, the encoder is responsible for reducing the coding, interpixel and psychovisual redundancies of input image. In first stage, the mapper transforms the input image into a format designed to reduce interpixel redundancies. The second stage, qunatizer block reduces the accuracy of mapper’s output in accordance with a predefined criterion. In third and final stage, a symbol decoder creates a code for quantizer output and maps the output in accordance with the code. These blocks perform, in reverse order, the inverse operations of the encoder’s symbol coder and mapper block. As quantization is irreversible, an inverse quantization is not included.

6. Types of Compression

The following section takes a closer look at lossy and loss less compression techniques. (Gaurav, Sanjay, & Rajeev, October 2013)

In the case of video, compression ratio causes some information to be lost ; some information at a detailed level is considered not essential for a reasonable reproduction of scene. This type of compression is called lossy compression, audio compression on the other hand is not lossy, and it is called loss less compression.

6.1 Lossless Compression

In lossless compression scheme, the reconstructed image after compression is numerically identical to original image. However lossless compression can only achieve a modest amount of compression. Lossless coding guaranties that, the decompressed image to be absolutely identical to the image before compression.

This is an important requirement for some application domain, e.g. medical imaging, where not only high quality is in demand but unaltered archiving is a legal requirement. Lossless technique can also be used for the compression of other data types where loss of information is not acceptable, e.g. text document and program executable.

6.2 Lossy Compression

Lossy is a term applied to data compression technique in which some amount of the original data is lost during the compression process. Lossy image compression applications attempt to eliminate redundant or unnecessary information in terms of what the human eye can perceive. An image reconstructed following lossy compression contains degradation relative to the original image. Often this is because the compression scheme completely discards redundant information. However lossy scheme are capable of achieving much higher compression. Under normal viewing condition, no visible loss is perceived (visually lossless). Lossy image data compression is useful for application to the world wide images for quicker transfer across the internet. An image reconstructed following lossy compression contains degradation relative to the original. Often this is because the compression scheme completely discards the redundant information.

7. Applications of Compression

Applications of data compression are primarily in transmission and in storage of information. Image transmission application are in broadcast television remote sensing via satellite, military application via aircraft, radar and sonar, teleconferencing, computer communication, facsimile transmission, etc. Image storage is required for educational business documents, medical images, that rises in computer tomography, magnetic resonance imaging and digital radiology, motion picture, satellite image weather maps, etc. Application of data compression is also possible in the development of fast algorithm, where the number of operations required to be implement an algorithm is reduced by working with compressed data. Over the year, the need for image compression has grown steadily. Currently it is recognized as an Enabling technology. It plays a crucial role in many important and diverse applications such as:

- Business document, where lossy compression is prohibited for legal reasons.
- Satellite image where the data loss is undesirable because of image collect cost.
- Medical image where difference in original image and uncompressed one can compress diagnostic accuracy.
- Tele -video conferencing.
- Remote sensing.
- Space and hazardous waste water application
- Facsimile transmission (fax)

8. Characteristic to Judge Compression Algorithm

Image quality describes the fidelity with which an image compression scheme recreates the source image data. There are four main characteristics to judge image compression algorithms

- Compression Ratio
- Compression Speed
- Mean Square Error
- Peak Signal to Noise Ratio

These characteristics are used to determine the suitability of a given compression algorithm for any application.

8.1 Compression Ratio

The compression ratio is equal to the size of the original image divided by the size of the compressed image by the following formula (KODITUWAKKU & AMARASINGHE, 2015):

$$CR = \frac{uncompressed\ size}{compressed\ size}$$

This ratio gives how much compression is achieved for a particular image. The compression ratio achieved usually indicates the picture quality. Generally, the higher the compression ratio, the poorer the quality of the resulting image. The tradeoff between compression ratio and picture quality is an important one to consider when compressing images. Some compression schemes produce compression ratios that are highly dependent on the image content. This aspect of compression is called data dependency. Using an algorithm with a high degree of data dependency, an image of a crowd at a football game (which contains a lot of detail) may produce a very small compression ratio, whereas an image of a blue sky (which consists mostly of constant colors and intensities) may produce a very high compression ratio.

8.2 Compression Speed

Compression time and decompression time are defined as the amount of time required to compress and decompress a picture, respectively. Their value depends on the following considerations:

- The complexity of the compression algorithm.
- The efficiency of the software or hardware implementation of the algorithm.
- The speed of the utilized processor or auxiliary hardware.

Generally, the faster that both operations can be performed the better. Fast compression time increases the speed with which material can be created. Fast decompression time increases the speed with which the user can display and interact with images (KODITUWAKKU & AMARASINGHE, 2015).

8.3 Mean Square Error

Mean square error measures the cumulative square error between the original and the compressed image. The formula for mean square is given as (Yusra & Chen, 2012):

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2$$

Where *m*n* is the size of the image, *I(i, j)* and *K(i, j)* are the matrix element of the decompressed and the original image at *(i, j)* pixel.

8.4 Peak Signal to Noise Ratio

The measure of peak signal-to-noise ratio (PSNR) is defined as the following formula:

$$PSNR = 10log_{10}\left(\frac{s^2}{MSE}\right)(dB)$$

Where s = 255 for an 8-bit image. The PSNR is basically the SNR when all pixel values are equal to the maximum possible value (Yusra & Chen, 2012).

SECTION 2: IMAGE COMPRESSION TECHNIQUES

On the bases of our requirements image compression techniques are broadly bifurcated in following two major categories.

- Lossless image compression
- Lossy image compression

1. Lossless Compression Techniques

Lossless compression compresses the image by encoding all the information from the original file, so when the image is decompressed, it will be exactly identical to the original image. There are several algorithms and techniques proposed in this filed, we can present the most popular one.

1.1 RLE Compression

The RLE compression method (Run Length Encoding, sometimes written as RLC for Run Length Coding) is used by many image formats (BMP, PCX, TIFF). It is based on the repetition of consecutive elements. This algorithm deals with some sort of redundancy (Blelloch, 2002).

The basic principle consists in coding a first element by giving the number of repetitions of a value and then the value to be repeated. Thus, according to this principle, chain “AAAAAHHHHHHHHHHHHHH” when compressed yields “5A1 4H”. The compression gain is thus (1 9-5) /1 9, that is, approximately 73.7% On the other hand, for the chain “CORRECTLY”, where there is little character repetition, the result of the compression is “1 C1 O2R1 E1 C1 T1 L1 Y”, thus compression proves to be very expensive here, with a negative compression gain of (9-1 6)/9 that is, -78%!

Actually, RLE compression is governed by particular rules which allow compression to be carried out when necessary and the chain to be left as it is when compression causes a waste. These rules are the following:

- If three or more elements are repeated consecutively, the RLE compression method is used
- If not, a control character (00) is inserted, followed by the number of elements of the non-compressed chain and then the latter
- If the number of elements of the chain is odd, the control character (00) is added on the end
- Finally, specific control characters were defined in order to code:
    - an end of line (00 01)
    - the end of the image (00 00)
    - A pointer displacement over the image of XX columns and YY rows in the reading direction (00 02 XX YY).

Thus, RLE compression makes no sense except for data with many consecutive repeated elements, in particular images with large uniform areas. This method however has the advantage of not being very difficult to implement. There are alternatives in which the image is encoded by blocks of pixels, in rows, or even in zigzag.

1.2 Huffman Encoding

In computer science and information theory, Huffman coding is an entropy encoding algorithm used for lossless data compression. It was developed by Huffman. Huffman coding today is often used as a "back-end" to some other compression methods (Mridul, Seema, & Dheeraj, 2012). Huffman coding works in a very similar manner to Shannon-Fano Coding, but the binary tree is built from the top down to generate an optimal result.

The algorithm to generate Huffman codes shares its first steps with Shannon-Fano (HUFFMAN, 1952):

1. Parse the input, counting the occurrence ofeach symbol.
2. Determine the probability ofeach symbol using the symbol count.
3. Sort the symbols by probability, with the most probable first.
4. Generate leaf nodes for each symbol, including P, and add them to a queue.
5. While (Nodes in Queue > 1)
   - Remove the two lowest probability nodes from the queue.
   - Prepend 0 and 1 to the left and right nodes' codes, respectively.
   - Create a new node with value equal to the sum of the nodes' probability.
   - Assign the first node to the left branch and the second node to the right branch.
   - Add the node to the queue
6. The last node remaining in the queue is the root of the Huffman tree

1.3 Arithmetic Coding

This method was developed in 1 979 at IBM, which was investigating data compression techniques for use in their mainframes. Arithmetic coding is arguably the most optimal entropy coding technique if the objective is the best compression ratio since it usually achieves better results than Huffman Coding. It is, however, quite complicated compared to the other coding techniques (RISSANEN & LANGDON, 1979).

Rather than splitting the probabilities of symbols into a tree, arithmetic coding transforms the input data into a single rational number between 0 and 1 by changing the base and assigning a single value to each unique symbol from 0 up to the base. Then, it is further transformed into a fixed-point binary number which is the encoded result. The value can be decoded into the original output by changing the base from binary back to the original base and replacing the values with the symbols they correspond to.

A general algorithm to compute the arithmetic code is:

1. Calculate the number of unique symbols in the input. This number represents the base b (e.g. base 2 is binary) of the arithmetic code.
2. Assign values from 0 to b to each unique symbol in the order they appear.
3. Using the values from step 2, replace the symbols in the input with their codes
4. Convert the result from step 3 from base b to a sufficiently long fixed-point binary number to preserve precision.
5. Record the length of the input string somewhere in the result as it is needed for decoding.

Here is an example of an encode operation, given the input “ABCDAABD”:

1. Found 4 unique symbols in input, therefore base = 4. Length = 8
2. Assigned values to symbols: A=0, B=1 , C=2, D=3
3. Replaced input with codes: “0.01230013_4 ” where the leading 0 is not a symbol.
4. Convert “0.01231123_4 ” from base 4 to base 2: “0.011011000001111_2”
5. Result found. Note in result that input length is 8.

Assuming 8-bit characters, the input is 64 bits long, while its arithmetic coding is just 15 bits long resulting in an excellent compression ratio of 24%. This example demonstrates how arithmetic coding compresses well when given a limited character set.

1.4 LZW Coding

LZW is the Lempel-Ziv-Welch algorithm created in 1984 by **Terry Welch**. It is the most commonly used derivative of the LZ78 family, despite being heavily patent encumbered. LZW improves on LZ78 in a similar way to LZSS; It removes redundant characters in the output and makes the output entirely out of pointers. It also includes every character in the dictionary before starting compression, and employs other tricks to improve compression such as encoding the last character of every new phrase as the first character of the next phrase. LZW is commonly found in the Graphics Interchange Format, as well as in the early specifications of the ZIP format and other specialized applications. LZW is very fast, but achieves poor compression compared to most newer algorithms and some algorithms are both faster and achieve better compression (Bell, Witten, & Cleary, 1989).

2. Lossy Compression Methods

Lossy compression techniques involve some loss of information, and data cannot be recovered or reconstructed exactly. In some applications, exact reconstruction is not necessary. For example, it is acceptable that a reconstructed video signal is different from the original as long as the differences do not result in annoying artifacts. However, we can generally obtain higher compression ratios than is possible with lossless compression. Lossy image compression techniques include following schemes:

2.1 Vector Quantization

Vector Quantization (VQ) is a lossy compression method. It uses a codebook containing pixel patterns with corresponding indexes on each of them. The main idea of VQ is to represent arrays of pixels by an index in the codebook. In this way, compression is achieved because the size of the index is usually a small fraction of that of the block of pixels.

The main advantages of VQ are the simplicity of its idea and the possible efficient implementation of the decoder. Moreover, VQ is theoretically an efficient method for image compression, and superior performance will be gained for large vectors. However, in order to use large vectors, VQ becomes complex and requires many computational resources (e.g. memory, computations per pixel) in order to efficiently construct and search a codebook. More research on reducing this complexity has to be done in order to make VQ a practical image compression method with superior quality (Sayood, 2000).

2.2 Predictive Coding

Predictive coding has been used extensively in image compression. Predictive image coding algorithms are used primarily to exploit the correlation between adjacent pixels. They predict the value of a given pixel based on the values of the surrounding pixels. Due to the correlation property among adjacent pixels in image, the use of a predictor can reduce the amount of information bits to represent image.

This type of lossy image compression technique is not as competitive as transform coding techniques used in modern lossy image compression, because predictive techniques have inferior compression ratios and worse reconstructed image quality than those of transform coding (Sayood, 2000).

2.3 Fractal Compression

The application of fractals in image compression started with M.F. Barnsley and A. Jacquin (Barnsley & et.al, 1988). Fractal image compression is a process to find a small set of mathematical equations that can describe the image. By sending the parameters of these equations to the decoder, we can reconstruct the original image.

In general, the theory of fractal compression is based on the contraction mapping theorem in the mathematics of metric spaces. The Partitioned Iterated Function System (PIFS), which is essentially a set of contraction mappings, is formed by analyzing the image. Those mappings can exploit the redundancy that is commonly present in most images. This redundancy is related to the similarity of an image with itself, that is, part A of a certain image is similar to another part B of the image, by doing an arbitrary number of contractive transformations that can bring A and B together. These contractive transformations are actually common geometrical operations such as rotation, scaling, skewing and shifting. By applying the resulting

PIFS on an initially blank image iteratively, we can completely regenerate the original image at the decoder. Since the PIFS often consists of a small number of parameters, a huge compression ratio (e.g. 500 to 1000 times) can be achieved by representing the original image using these parameters. However, fractal image compression has its disadvantages. Because fractal image compression usually involves a large amount of matching and geometric operations, it is time consuming. The coding process is so asymmetrical that encoding of an image takes much longer time than decoding.

2.4 Transform Based Image Compression

The basic encoding method for transform based compression works as follows:

1. Image transform: Divide the source image into blocks and apply the transformations to the blocks.
2. Parameter quantization: The data generated by the transformation are quantized to reduce the amount of information. This step represents the information within the new domain by reducing the amount of data. Quantization is in most cases not a reversible operation because of its lossy property.
3. Encoding: Encode the results of the quantization. This last step can be error free by using Run Length encoding or Huffman coding. It can also be lossy if it optimizes the representation of the information to further reduce the bit rate.

Transform based compression is one of the most useful applications. Combined with other compression techniques, this technique allows the efficient transmission, storage, and display of images that otherwise would be impractical (Gu.H, 2000).

2.5 DCT-Based Transform Coding

The Discrete Cosine Transform (DCT) (G.Strang, 1999) was first applied to image compression in the work by **Ahmed**, **Natarajan**, and **Rao**. It is a popular transform used by the JPEG (Joint Photographic Experts Group) image compression standard for lossy compression of images. Since it is used so frequently, DCT is often referred to in the literature as JPEG-DCT, DCT used in JPEG.

2.6 JPEG-DCT

JPEG-DCT is a transform coding method comprising four steps. The source image is first partitioned into sub-blocks of size 8x8 pixels in dimension. Then each block is transformed from spatial domain to frequency domain using a 2-D DCT basis function. The resulting frequency coefficients are quantized and finally output to a lossless entropy coder (Wallace & et.al, 1991).

Lossy JPEG encoding process is shown in Figure 18.

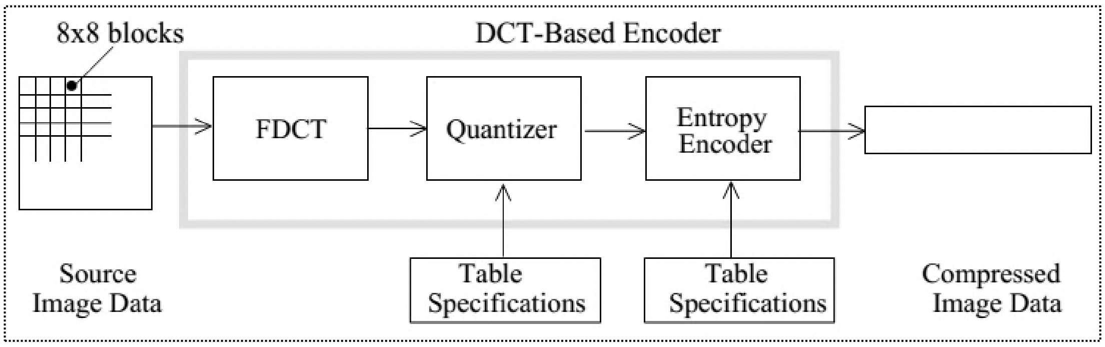


Figure 18: DCT-Based Encoder Processing Steps

Lossy JPEG decoder process is shown in Figure 19.

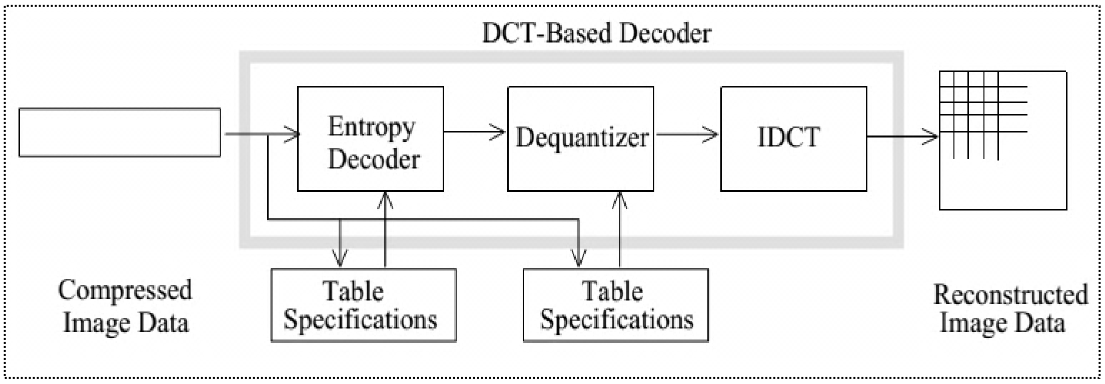


Figure 19: DCT-Based Decoder Processing Steps

CONCLUSION

In this chapter we have presented the general scheme of image compression and different types of image compression techniques. As a summary, image compression is an application of data compression that attempts to reduce the redundancy of the image to be stored or transmit in an efficient form. Basically image compression techniques classified into two: Lossy compression techniques and lossless compression techniques. As the name indicates Lossless compression reduces the number of bits required to represent an image such that the reconstructed image is numerically identical to the original one on a pixel-by-pixel basis. On the other hand, lossy compression schemes allow degradations in the reconstructed image in exchange for a reduced bit rate. These degradations may or may not be visually apparent, and greater compression can be achieved by allowing more degradation.

CHAPTER 3
LOGARITHMIC FUNCTIONS FOR IMAGE COMPRESSION: A MATHEMATICAL BACKGROUND and RELATED WORKS

INTRODUCTION

Since the last decade, there is an enormous amount of multimedia communication in integrated data services such as Internet, Mobile multimedia, data sensing in sensor networks, video conferencing, facsimile transmission, document and medical imaging using voice, text, image and video (Huang, Chen, & Chang, 1999). In image communication, images are compressed and transmitted. Image compression is necessary for one or more reasons such as reduction of storage space, reduction of transmission time and bandwidth constraint. The important requirements of any compression technique are high CR, high quality and high speed (Huang, Chen, & Chang, 1999).

Applied mathematics is the branch of mathematics that consists of the application of mathematical knowledge to other fields, like the use of mathematics in business, medicine, industry, …etc. as the title indicates logarithmic functions are mathematical functions which have some properties that can be considered useful in image compression.

In this chapter, we will present a mathematical background and we will discuss works related to LOG-EXP based image compression.

SECTION 1: A MATHEMATICAL BACKGROUND OF LOGARITHMIC FUNCTIONS

1. Introduction

In this section we present a mathematical background of logarithmic functions as follows: first, we introduce the logarithmic functions and how to apply these functions in image compression. Then, we give a general mathematical presentation of these functions and its inverse definition, derivation …etc. Finally, we explain why these functions are useful in image compression.

1. What is Logarithmic Function?

In mathematics, the logarithm is the inverse operation to exponentiation. It means the logarithm of a number is the exponent to which another fixed value; the base must be raised to produce that number. In simple cases the logarithm counts repeated multiplication. For example, the base 10 logarithm of 1000 is 3, as 10 to the power 3 is 1000 (1000 = 10 × 10 × 10 = 103); the multiplication is repeated three times. More generally, exponentiation allows any positive real number to be raised to any real power, always producing a positive result, so the logarithm can be calculated for any two positive real numbers b and x where b is not equal to 1. The logarithm of x to base b, denoted $\log_b$ (x), is the unique real number y such that $b^y=x$.

An example of the use of these functions is as follows: we consider the values set bellow:

A = {64 128 128 32 64 256 16 256} to store these values on the disc we need 64 bits (8 bits for its one).

We transform A by applying the log_2 function, we get:

B = {6 7 7 5 6 8 4 8} to store these values on the disc we need only 32 bits(4 bits for its one).To return to A from B we can apply the *exp_2* function.

2. Exponential and Logarithmic Functions

2.1 Exponential function

Exponentiation**:** The third power of number *b* is the product of three factors of *b*. More generally, raising *b* to the *n*-th power, where *n* is a natural number, is done by multiplying *n* factors of *b*. The *n*-th power of *b* is written b^n, so that

$$b^n = \underbrace{b \times b \times \cdots \times b}_{n \text{ factors}}.$$

Exponentiation may be extended to b^y, where *b* is a positive number and the *exponent y* is any real number. For example, b^{-1} is the reciprocal of *b*, that is, 1/*b* (Shirali, 2002).

Definition of an Exponential Function: For $a > 0$, where $a \neq 1$, the exponential function with base *a* is defined by $f(x) = a^X$ (or equivalently $y = a^X$), where "*a*" is called the base, and "*x*" is called the "exponent".

The Natural Exponential Function: There is an irrational number, denoted *e*, which arises in many logarithmic and exponential function problems as a base. The value of *e* is approximately 2.71828…. The natural exponential function is $f(x)= e^X$ (or equivalently $y = e^X$).

The exponential function ex can be characterized in a variety of equivalent ways. In particular it may be defined by the following power series (Rudin, 1987):

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$$

It is also the following limit:

$$e^x = \lim_{n\to\infty} \left(1 + \frac{x}{n}\right)^n$$

The natural exponential graph**:**

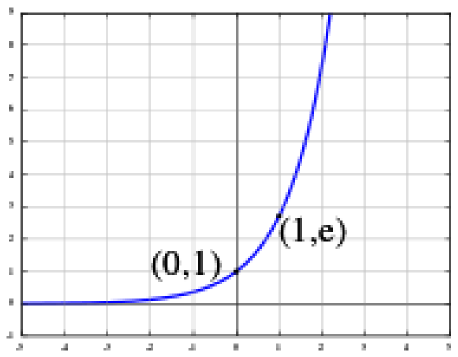


Figure 20: The natural exponential function y = e^{x}

Derivative equation**:**

The importance of the exponential function in mathematics and the sciences stems mainly from properties of its derivative. In particular,

$$\frac{\mathrm{d}}{\mathrm{d}x}e^x = e^x$$

The Natural Exponential Properties**:**

$$\forall x \in \mathbb{R} \quad \forall y \in \left]0;+\infty\right[\qquad e^x = y \Leftrightarrow x = \ln y$$

$$\forall x \in \mathbb{R} \quad \ln e^x = x \qquad \forall x \in \left]0;+\infty\right[\quad e^{\ln x} = x$$

$$\forall (a;b) \in \mathbb{R}^2 \quad \forall r \in \mathbb{Q} \quad e^{a+b} = e^a.e^b \quad e^{-a} = \frac{1}{e^a} \quad e^{a-b} = \frac{e^a}{e^b} \quad e^{rb} = \left(e^a\right)^r$$

$$\forall (a;b) \in \mathbb{R}^2 \quad e^a = e^b \Leftrightarrow a = b$$

2.2 Logarithmic Functions

Motivation**:** The idea of logarithms is to reverse the operation of exponentiation, which is raising a number to a power. For example, the third power (or cube) of 2 is 8, because 8 is the product of three factors of 2:

$$2^3 = 2 \times 2 \times 2 = 8.$$

It follows that the logarithm of 8 with respect to base 2 is 3, so log2 8 = 3.

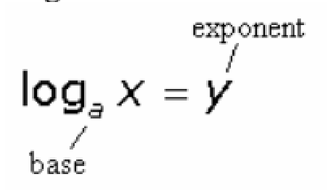
Definition of the Logarithmic Function**:** For $a > 0$ with $a \neq 1$, the logarithmic function with base a, denoted log_a , is defined by $log_a\, x = y$ if and only if $a^y = x$.

In other words, the logarithm of x to base b is the solution y to the equation (Kate & Bhapkar, 2009):

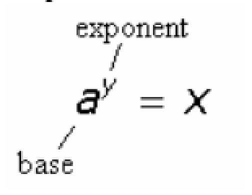
$$b^y = x$$

You can easily switch between logarithmic form and exponential form as follows:

Logarithmic form



Exponential form



The natural logarithm: The logarithmic function with base e is called the natural logarithm and is denoted by the special notation: $log_e\, x = ln\, x$

Examples**:**

For example, $\log_2 (16) = 4$, since $24 = 2 \times 2 \times 2 \times 2 = 16$. Logarithms can also be negative:

$$\log_2\left(\frac{1}{2}\right) = -1,$$

Since $$2^{-1} = \frac{1}{2^1} = \frac{1}{2}.$$

A third example: $\log_{10} (150)$ is approximately 2.176, which lies between 2 and 3, just as 150 lies between $10^2 = 100$ and $10^3 = 1000$. Finally, for any base b, $\log_b(b) = 1$ and $\log_b(1) = 0$, since $b^1 = b$ and $b^0 = 1$, respectively.

Change of base**:**

The logarithm $\log_b(x)$ can be computed from the logarithms of x and b with respect to an arbitrary base k using the following formula:

$$\log_b(x) = \frac{\log_k(x)}{\log_k(b)}.$$

Typical scientific calculators calculate the logarithms to bases 10 and *e*. (Bernstein, 1999) Logarithms with respect to any base *b* can be determined using either of these two logarithms by the previous formula:

$$\log_b(x) = \frac{\log_{10}(x)}{\log_{10}(b)} = \frac{\log_e(x)}{\log_e(b)}.$$

Given a number *x* and its logarithm $\log_b$(x) to an unknown base *b*, the base is given by:

$$b = x^{\frac{1}{\log_b(x)}}.$$

The natural logarithmic graph**:**

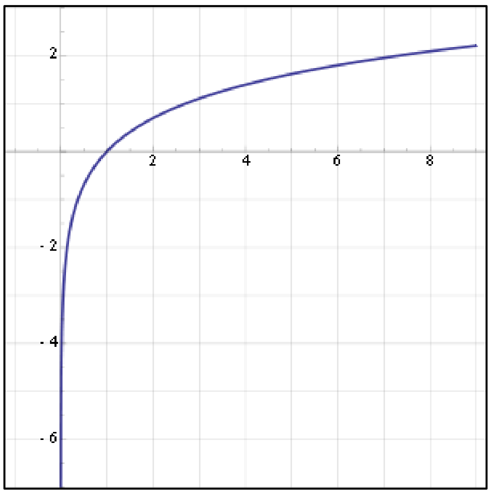


Figure 21: Graph of the natural logarithm function

Derivative equation**:**

The derivative of the natural logarithm is given by

$$\frac{d}{dx}\ln(x) = \frac{1}{x}.$$

The logarithmic function Properties**:**

$$log_a(MN) = \ log_aM + log_aN$$

$$log_a \frac{M}{N} = log_a M - log_a N$$

$$log_a N^k = k \, log_a N$$

$$log_a \sqrt[k]{N} = \frac{1}{k} log_a N$$

3. Applications

Logarithms have many applications inside and outside mathematics. Some of these occurrences are related to the notion of scale invariance. For example, each chamber of the shell of a nautilus is an approximate copy of the next one, scaled by a constant factor. This gives rise to a logarithmic spiral (Maor, 2009). Benford's law on the distribution of leading digits can also be explained by scale invariance (Frey, 2006). Logarithms are also linked to self-similarity. For example, logarithms appear in the analysis of algorithms that solve a problem by dividing it into two similar smaller problems and patching their solutions (Ricciardi, 1990). The dimensions of self-similar geometric shapes, that is, shapes whose parts resemble the overall picture are also based on logarithms. Logarithmic scales are useful for quantifying the relative change of a value as opposed to its absolute difference. Moreover, because the logarithmic function log(x) grows very slowly for large *x*; logarithmic scales are used to compress large-scale scientific data. Logarithms also occur in numerous scientific formulas, such as the Tsiolkovsky rocket equation, the Fenske equation, or the Nernst equation.

3.1 Application domains

- Logarithmic scale
- Probability theory and statistics
- Computational complexity
- Entropy and chaos
- Fractals
- Number theory

4. Conclusion

Logarithmic functions have some properties that can be considered useful in data compression, especially in lossy scheme, when losing some information of data like audio, video, and even images, especially in applications such as streaming media. Lossy transforms based image compression; it uses this function in first step to transform the pixels values to the coefficients, in other words, it prepares the image to quantization phase.

SECTION 2: RELATED WORKS

1. Introduction

The JPEG still image compression standard (Wallace & et.al, 1991) satisfies these basic requirements and is widely used in many multimedia applications. The main drawbacks of JPEG are block artifacts and low throughput (Han & Leou, April 1998). To overcome the above drawbacks in JPEG, S.C.Huang,L.G.Chen and H.C.Chang (Huang, Chen, & Chang, 1999) proposed a LOG-EXP based still image compression algorithm. The speed of this algorithm is limited because of the presence of Huffman coding block.

Natarajan Somasundaram, and Y.V. RamanaRao in (Natarajan& Ramana, September 2008) proposed Modified LOG-EXP Based Image Compression Algorithm. In this novel algorithm two modifications are carried out in the algorithm of (Huang, Chen, & Chang, 1999) to reduce the compression time without sacrificing CR and image quality.

2. A Novel Image Compression Algorithm by Using LOG-EXP Transform

In (Huang, Chen, & Chang, 1999)Huang et.al proposed a novel VLSI architecture based on LOG-EXP compression to satisfy the high speed requirement and avoid the block artifact of the high quality image compression.

2.1 Algorithm

Based on the logarithmic number system (LNS) properties, the LOG-EXP image compression system (Huang & Chen, 1998) is proposed. The compression and decompression flow is shown on Figure 22. For the gray level image, the pixel value is usually represented in integer format. Due to the finite word length effect and the continuous distribution of the pixel value, the original 8bits gray level image can be represented in 7 bits LNS format. In the 7bits fixed point LNS format, the integer part is3 bits and the fraction is 4 bits. The logarithmic transform removes the redundancy of the data representation. At the same time, the large neighboring difference between the two pixels is reduced to small range in LNS format.

To compute the differences between the neighbor pixels, the neighboring differences will be very small and there are many similar differences. For removing the large neighboring difference from this line to the next line, the snake scan is used. Besides, the repeat reduction of the neighboring difference is also shown in Figure 23. After reducing the repeat items, the data part and times part are processed with the Huffman encoder to remove the redundancy among the large amount of similar arid regular neighboring differences.

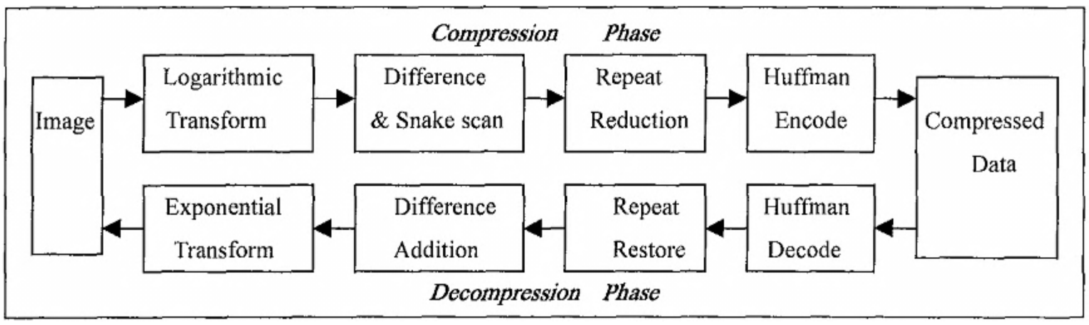


Figure 22: LOG-EXP Compression System

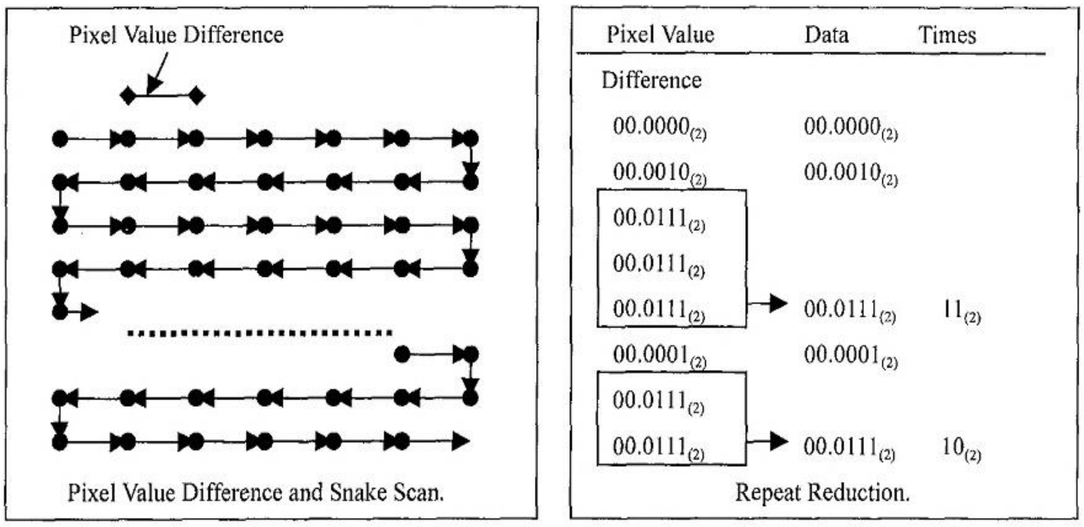


Figure 23: LOG-EXP Algorithm Design

2.2 Evaluation

The proposed algorithm in (Huang, Chen, & Chang, 1999) tested for gray level image and the results are given in the following table.

Table 5: Compression Ratio for gray level Benchmark

	Lena	Baboon	Pepper
CR	8.40	7.36	7.91
PSNR	41.52	41.31	41.51

2.3 Comparison with JPEG-Standard

Table 6: Comparison between LOG-EXP and JPEG for gray level benchmark

Image	measurements	JPEG	LOG-EXP
Pepper	CR	4.13	7.91
	PSNR	39.23	41.51
Lena	CR	4.57	8.40
	PSNR	41.23	41.52
baboon	CR	2.39	7.31
	PSNR	37.23	41.31

3. Modified LOG-EXP Based Image Compression Algorithm

In (Natarajan & Ramana, September 2008) Natarajan et.al proposed a modified LOG – EXP transform based image compression algorithm obtained by using arithmetic coding in the place of Huffman coding to reduce the computation time of the original LOG – EXP transform based algorithm.

3.1 Algorithm

Figure.3. shows the block diagram of the proposed compression and decompression system which is a modification of the algorithm of (Huang, Chen, & Chang, 1999). Two modifications are introduced in the proposed scheme as compared to the algorithm of (Huang, Chen, & Chang, 1999), namely, the level shifting of image data by 256 and usage of arithmetic coding in place of Huffman coding. The steps in the proposed algorithm are as follows:

1. Pixels are scanned using snake scan of (Huang, Chen, & Chang, 1999).
2. Each pixel is level shifted by adding a value equal to256.
3. Logarithm based on log2 is computed for each level shifted pixel to get log transformed image. The addition of 256 to each pixel causes log values of all the pixels to have a value 8 (1000)2 in the integer portion which does not need to be explicitly stored for processing. This is in contrast to the algorithm of (Huang, Chen, & Chang, 1999) where the integer part of the log values of the pixels varies between 0 and 7 which needs to be considered for processing. In this paper, only the fractional part is considered for processing. The fractional part of the log values are truncated to 3 decimal digits.
4. Repeat reduction block outputs are of the form (*xi, yi, and pi*) where xi is the log value of the pixel, *yi* is the number of continuous occurrences of xi and pi is the probability of occurrence of the pattern (*xi, yi*) in the entire image.

5. A codebook with index is created where each entry consists of (*xi, yi, and pi*). The maximum number of entries in the codebook is 256.

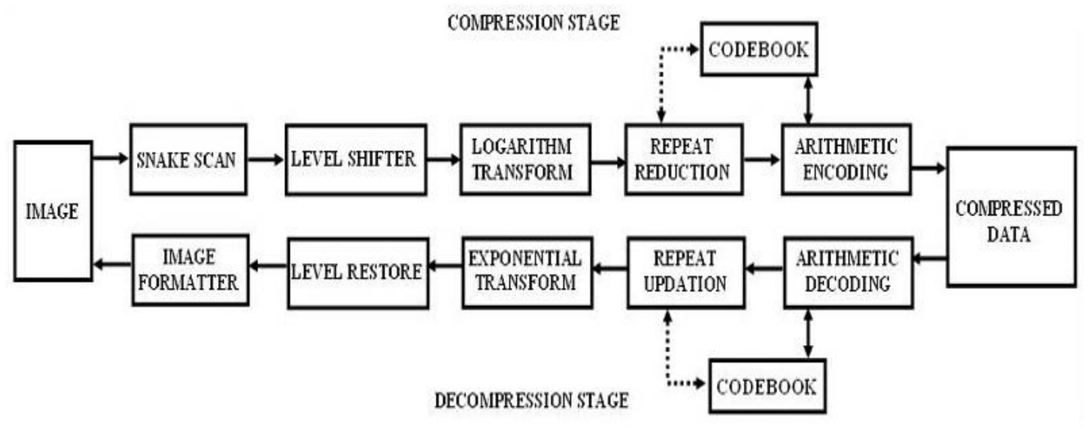


Figure 24: Proposed LOG-EXP Image Compression – Decompression System

3.2 Evaluation

The proposed algorithm in (Natarajan & Ramana, September 2008) tested for gray level image and the results are given in the following table.

Table 7: Compression Ratio for gray level Benchmark

	Lena	Pepper	Baboon
CR	9.01	8.47	8.42
PSNR	42.36	42.31	42.30

3.3 Comparison with (Huang, Chen, & Chang, 1999)

For the purpose of comparison, three gray-scale images namely, Lena, Pepper and Mandrill are considered. Each image is of size 512 x 512 with 8 bit resolution.

The performance of the proposed algorithm is compared with that of algorithm of (Huang, Chen, & Chang, 1999) on the basis of PSNR, CR, subjective quality and execution time.

Tables 1, 2 and 3 show the values of PSNR, CR and execution times of the algorithm of (Huang, Chen, & Chang, 1999)[1]and the proposed algorithm for Lena, Pepper and Mandrill images respectively.

Table 8: Results for Lena image

PSNR (dB)	Algorithm of [1]		Proposed Algorithm	
	CR	Execution Time (s)	CR	Execution Time (s)
29.89	14.77	180.51	14.85	18.20
42.36	8.95	312.23	9.01	29.67
50.68	6.52	592.12	6.54	39.88
62.19	5.95	692.62	5.96	43.64

Table 9: Results for Pepper image

PSNR (dB)	Algorithm of [1]		Proposed Algorithm	
	CR	Execution Time (s)	CR	Execution Time (s)
29.71	13.78	237.34	13.86	19.33
42.31	8.44	386.36	8.47	31.67
50.95	6.24	735.97	6.26	41.81
61.91	5.77	825.08	5.79	44.69

[1] In the following tables is denoted by Algorithm of [1]

Table 10: Results for Baboon image

PSNR (dB)	Algorithm of [1]		Proposed Algorithm	
	CR	Execution Time (s)	CR	Execution Time (s)
29.65	12.37	220.29	12.43	2.04
42.30	8.38	392.17	8.42	31.99
50.93	6.47	766.44	6.49	40.76
63.55	5.91	847.29	5.93	43.80

4. Conclusion

In this section we have presented two related works that use logarithmic functions in image compression. LOG-EXP based image compression technique is proposed in (Huang, Chen, & Chang, 1999) to yield reconstructed images with better subjective quality than that obtained using JPEG standard. This method was modified in (Natarajan & Ramana, September 2008) by using arithmetic coding in place of Huffman coding. It is found that this improves the speed of execution of the LOG-EXP based image compression algorithm without altering the PSNR, CR and subjective quality characteristics of the original LOG-EXP based image compression algorithm.

CONCLUSION

In this chapter we have presented mathematically the logarithmic functions and its inverse the exponential functions with some properties of both and their domains application in section one. In section 2, we have presented two related works which are: A Novel Image Compression Algorithm by Using LOG-EXP Transform; and Modified LOG-EXP Based Image Compression Algorithm.

These two works either (Huang, Chen, & Chang, 1999) and (Natarajan & Ramana, September 2008) are proposed for gray level image and have a common limit in reason of execution time. On the other hand, the LOG-EXP image compression can get high compression ratio for the complex texture image (e.g. benchmark image baboon) and the high quality image.

CHAPTER 4
IMAGE COMPRESSION USING LOGARITHMIC FUNCTIONS WITH PIXELS SHIFTING

INTRODUCTION

Nowadays, image compression techniques are very common in a wide area of researches. Two types of image compression have been introduced, which are lossless and lossy compression. Unlike lossless kind, the lossy image compression allows to lose some information at reason to achieve better compression ratio as much as possible, with acceptable degradation in quality of the reconstructed image.

The use of a capable transform to detect the essential information through an image, may allow representing it by a few numbers of significant coefficients, thereby reducing the numbers of bits required to represent the compressed image at the end of the compression phase.

In this chapter, we present our proposed method for still image compression, which is based on logarithmic functions transform with pixels shifting to generate more redundancy in the image and prepared it to the quantization step which is based on the logarithmic scalar quantization.

SECTION 1: METHODOLOGY

1. Introduction

An efficient data compression can be summarized in three principal steps: first, reduce the correlation between pixels (by applying a transformation); second, the quantization; and last one is the entropy coding.

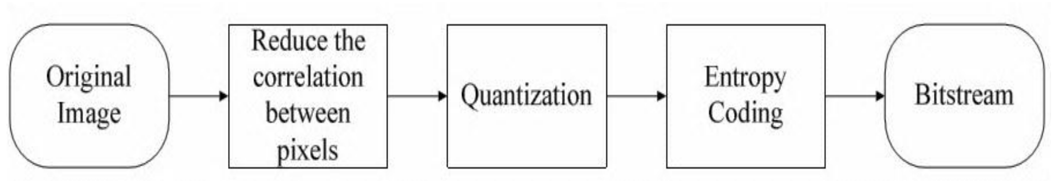


Figure 25: The general encoding flow of image compression

Our proposed method follows this general scheme of lossy image compression, and it has been proposed in order to achieve the following requirements:

- High compression ratio
- High quality of the decompressed image
- High computation speed
- Global method (not specified for a kind of image: ships, text, etc.)

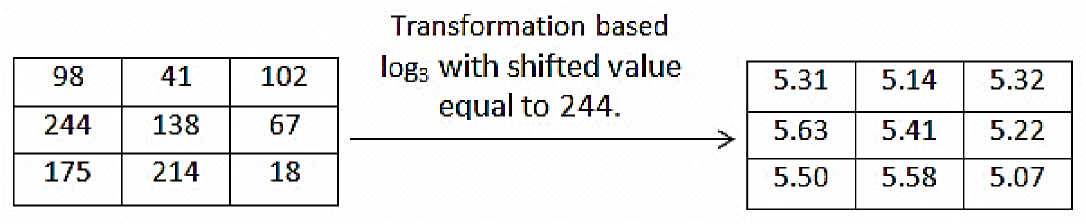
2. Outline Method

In several images, the values of one pixel and its adjacent pixels are very similar. Once the correlation between the pixels is reduced, we can take advantage of the basic data compression (lossless techniques) to reduce the storage quantity. Thus, finding a less correlated representation of an image is the most important part of the image compression algorithm.

Our proposed technique which is called logarithmic transform based image compression with pixels shifting (shortly LTBICPS) consists of the transformation based on the logarithmic numbers system (LNS) properties. The compression and decompression flow are shown in figure 26, figure 27, respectively.

For the color images, an image is formed by three components (matrixes), **R** for red color, **G** for green color, and **B** for blue color. In each one of these components, the value of one element is usually represented in integer format (i.e. 8 bits). The original 8 bits can represent in few bits by applied the logarithmic transform. The logarithmic transform is used in aim to remove the redundancy of the data representation. At the same time, the large neighboring difference between the two pixels is reduced to small range in logarithmic numbers system (LNS) format.

In our proposed method, we note that the logarithmic functions have a key property; the integer parts are the same in interval of two successive powers of the base, e.g., for base 2, the logarithm of [256; 511] is always 8. For this reason, each pixel is shifted by adding a value adapted to the logarithmic function base. The added value to each pixel causes log values of all the pixels to have the same value in the integer part which does not need to be explicitly stored for processing. This is in contrast to the algorithm of (Huang, Chen, & Chang, 1999) where the integer part of the log values of the pixels varies between 0 and 7 which needs to be considered for processing. Hence, only the fractional part is considered for processing.


Transformation based $\log_3$ with shifted value equal to 244.

98	41	102
244	138	67
175	214	18

5.31	5.14	5.32
5.63	5.41	5.22
5.50	5.58	5.07

The next step is the quantization phase. At this stage we propose to use a non-uniform quantization by compression. For a signal source not uniformly distributed, we can use a non-linier function F(x) to convert it from another signal source with probability distributed function near to a uniform distribution. Next, the uniform quantization can easily and efficiently used.

For the decoding process, first, we apply the uniform de-quantization to reconstruct the values witch uniformly quantized. Next, the invers function $F^{-1}(x)$ is applied to product the final quantized values.

The last step is the entropy coding, where we use a lossless compression software namely **paq8pxd_v4**[1]**.**

[1] Downloaded from: http://encode.ru/forums/2-Data-Compression.

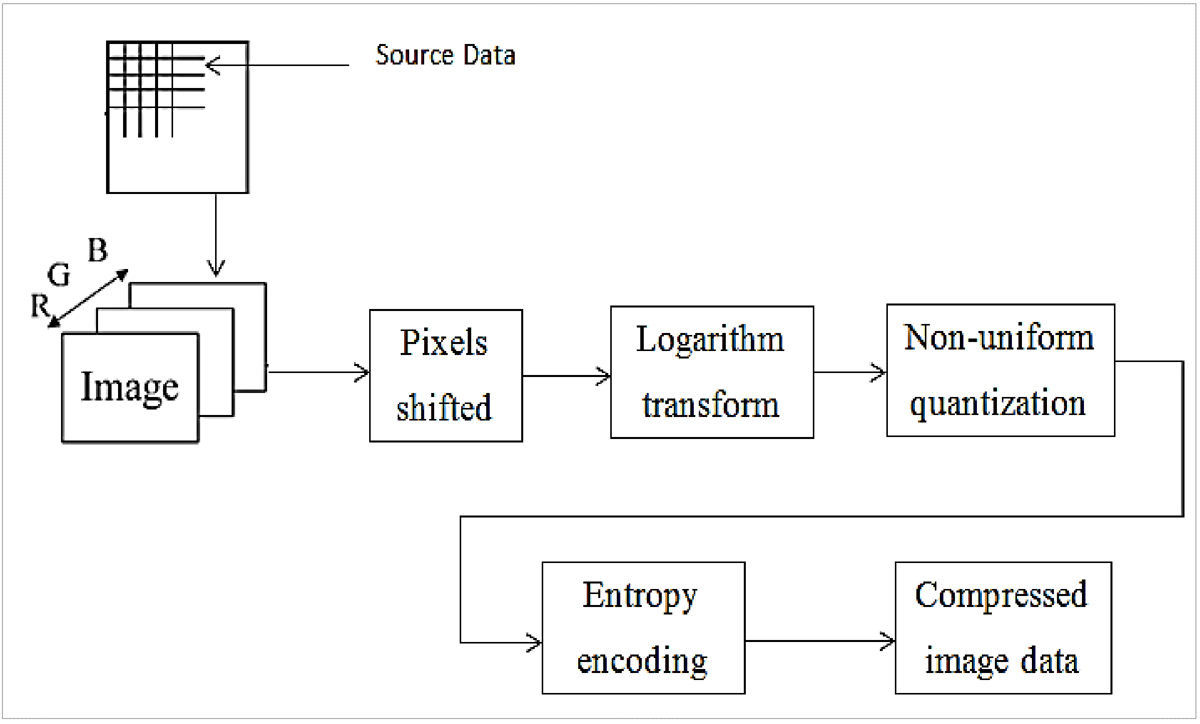


Figure 26: Logarithmic-Based Encoder Processing Steps

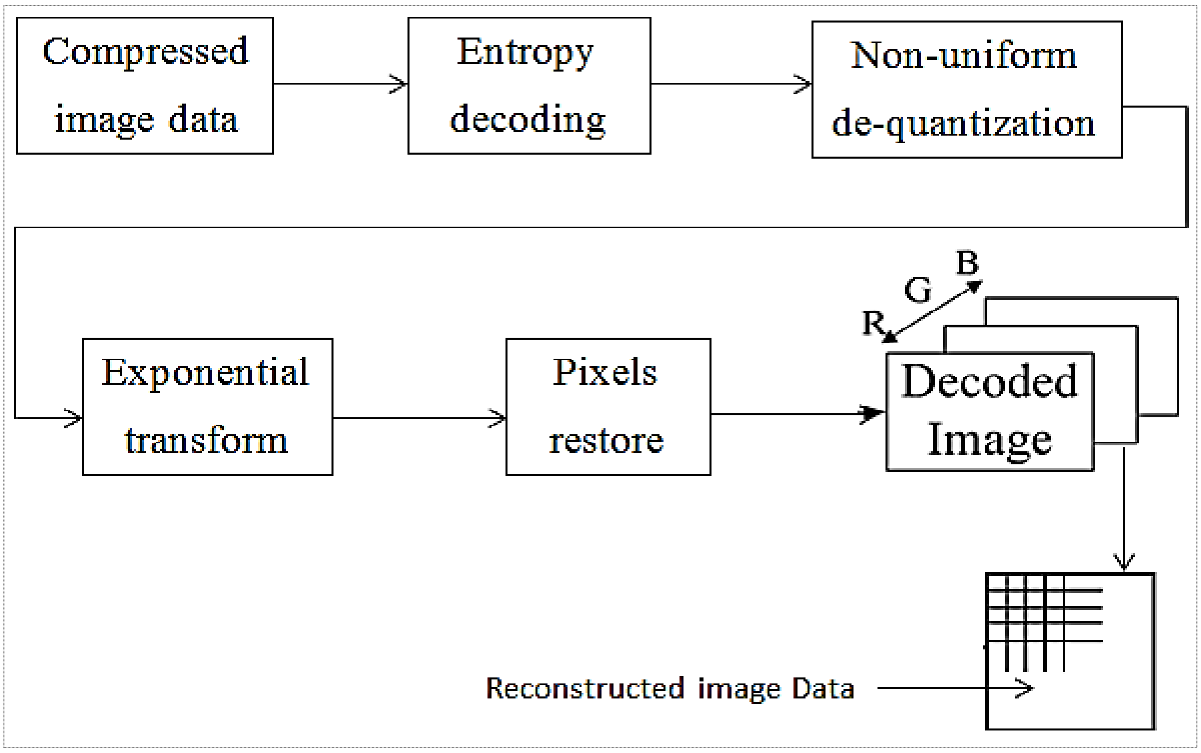


Figure 27: Exponential-Based Decoder Processing Steps

3. Architecture of the Proposed Method

3.1 The encoder model

The encoder model of proposed method consists of four steps shown in figure 26; the computing units are divided into the different phase and described in the following.

Pixels shifting: in the first steps, we read the source image data and divided it to three matrixes, one for red color, another one for green color, and one for blue color. Now, each pixel is shifted by adding a value adapted to the logarithmic function base. Next are some values adapted with some logarithmic bases:

Logarithmic function base 2: shifted value is 256 or 512, 1024…

Logarithmic function base 3: shifted value is 244 or 729…

Logarithmic function base 10: shifted value is 100 or 1000…

In pseudocode form, the pixel shifting is described by Algorithm 1.

Algorithm 1 Image reading and pixels shifting

```
1: read the source image in variable X from the specific file
2: keep the matrix of red color in variable R
3: keep the matrix of green color in variable G
4: keep the matrix of blue color in variable B
5: for each component R, G, and B do shift the pixels by adding the adapted value
```

Logarithmic-Based Transform: in this phase, the Logarithm based on $\log_b$ (b is the base, which can be 2, 3, 4…) is computed for each level shifted pixel to get log transformed image. The addition of shifted value to each pixel causes log values of all the pixels to have a same value in the integer portion which need not be explicitly stored for processing. Thus, we remove the integer part and keep only the fractional part for the quantization step. In following some integer part of some shifted value:

Log_2 with shifted value equal to 256 the integer part is 8

Log_2 with shifted value equal to 512 the integer part is 9

Log_3 with shifted value equal to 244 the integer part is 5

Log_3 with shifted value equal to 729 the integer part is 6

In pseudocode form, the logarithm transform is described by Algorithm 2.

Algorithm 2 the Logarithm-Based Transform

```
1: chose the logarithm function
2: for each component R, G, B do steps 3 to 5
3: apply the logarithm transform
4: eliminate the integer part for each transformed pixel
5: keep the fractional part to the quantization step
```

Quantization: After output from the logarithmic transform, each element is uniformly quantized in exponentiation with a choose base, which must be specified by the application (or user) as an input to the encoder. Each base can be any integer value from 2 to 255. The purpose of quantization is to achieve further compression by representing logarithm transform with no greater precision than is necessary to achieve the desired image quality. Stated another way, the goal of this processing step is to discard information which is not visually significant. Quantization is a many-to-one mapping, and therefore is fundamentally lossy.

Quantization is defined as exponentiation of each logarithm transform coefficient by the specified base (non-uniform quantization), followed by rounding to the nearest integer (uniform quantization):

In pseudocode form, the quantization step is described by Algorithm 3.

Algorithm 3 the quantization phase

```
1: do chose the exponentiation base (from 2 to 255)
2: for each logarithm transform output do apply the exponentiation
3: multiple by the precision value  (optional)
4:  round to the nearest integer
```

Entropy encoding: After output from the quantization step, we get a set of indexes, characterized by a high statistic redundancy caused by the large intervals of the quantization used.

The entropy encoding allows reducing the data size without introducing some distortion by exploiting the statistic redundancies.

To more benefit this phase and to gain more compression ratio, we use lossless compression software namely **paq8pxd_v4**.

In pseudocode form, the entropy encoding step is described by Algorithm 4.

Algorithm 4 Entropy encoding phase

```
1: regroup the quantization output of the 3 components in one 3D matrix
2: save it in the bmp format
3: compressed it by paq8pxd_v4.
```

3.2 The decoder model

The decoder model of proposed method consists also of four steps shown in figure 27, which are the inverse of the encoder process. The computing units are divided into the different phase and described in the following.

Entropy decoding: in this step we easily decompress the compressed data using the software paq8pxd_v4.

In pseudocode form, the entropy decoding step is described by Algorithm 5.

Algorithm 5 Entropy decoding phase

```
1: decompressed stored data by paq8pxd_v4.
2: recreate the 3 components R, G, and B from restored 3D matrix
3: reformed the quantization indexes of the 3 matrixes
```

De-quantization: the de-quantization will be easily the inverse one by one of the quantization phase. First, we apply the uniform de-quantization by dividing by the precision value. Next, we apply the non-uniform de-quantization by the inverse of the exponentiation (it is the logarithm), which the logarithm is based on the same base used for the exponentiation.

In pseudocode form, the de-quantization phase is described by Algorithm 6.

Algorithm 6 the de-quantization phase

```
1: do divide by the precision value  (if we multiple by the precision in the quantization phase)
2: do chose the logarithm base (the same base used for exponentiation)
3: for each element apply the logarithm based on the chosen base
```

Exponential transform**:** After output from the de-quantization, now, we apply the inverse transform of the logarithm transform, to do it, we use the exponential transform with the same base used of the logarithm-based transform.

In pseudocode form, the Exponential transform is described by Algorithm 7.

Algorithm 7 the Exponential-Based Transform

```
1: chose the exponentiation base
2: for each component R, G, B do steps 3 to 4
3: restore the integer part for each pixel by adding the removed value
4: apply the exponential transform
```

Pixels restore: in the last phase, we restore the initial pixels values, to do it; we decrease its pixel by a value equal to the value used to shift it.

In pseudocode form, the Exponential transform is described by Algorithm 8.

Algorithm 8 the Exponential-Based Transform

```
1: take the value used in the pixels shifting phase
2: for each component R, G, B do steps 3
3: restore the value for each pixel by decreasing it
```

4. Baseline Encoding Example

In [Appendix A], we give an example of Baseline compression and encoding of a single 8x8 sample block sample image.

SECTION 2: RESULTS AND DISCUSSION

1. Implementation

For the proposed method, the MATLAB implementation was performed on an Intel® Core™ i3-2310M CPU @ 2.10 GHz 2.10 GHz, 4.00 GB RAM, and the operating system platform is Microsoft Windows 7.

1.1 Graphic user interface description

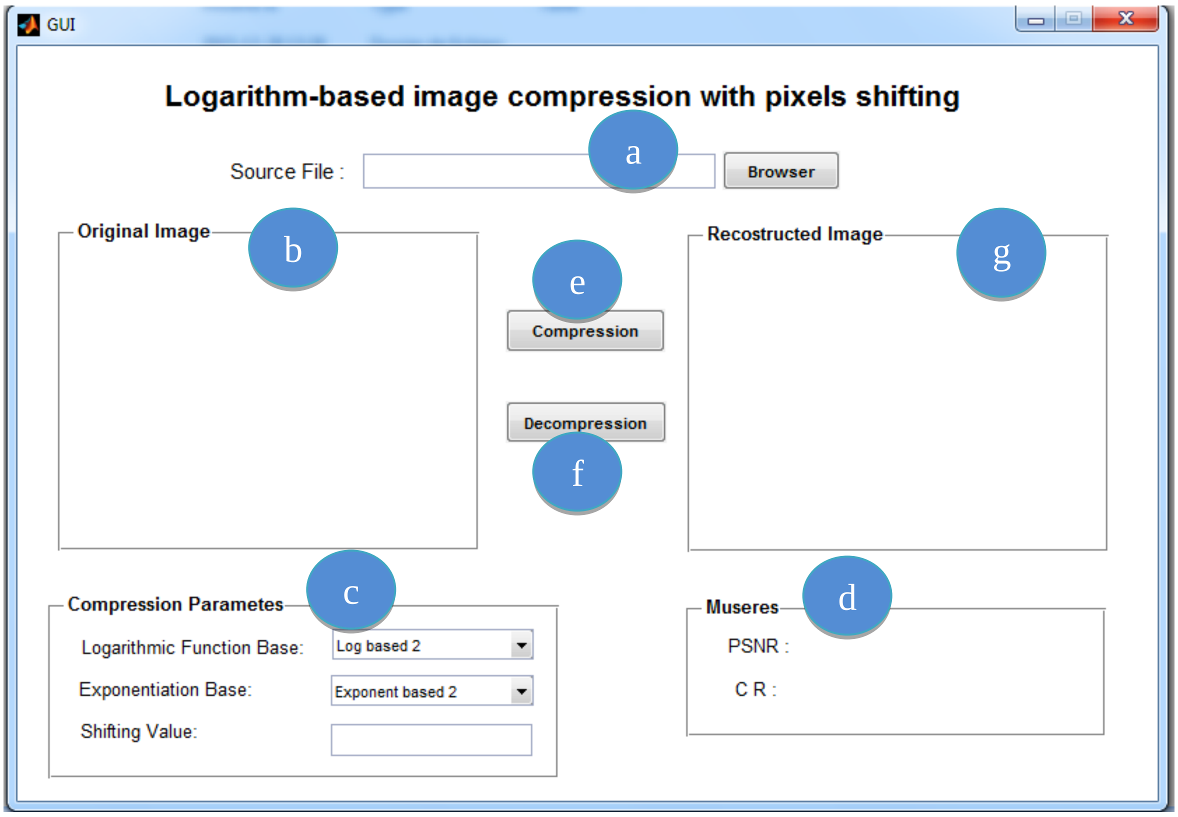


Figure 28: the GUI of our proposed method

a: shows the source file of the uncompressed image b: shows the loaded image which in question to compress

c: a set of parameters, which permit to adjust the application

d: shows the measurements of the compressed image

e: starts the compression processing

f: starts the decompression processing;

g: show the reconstructed image

2. Experimental Results

In this section, we illustrate the effectiveness of the proposed approach in image compression by presenting the results obtained from the experimentation.

Proposed method was implemented in MATLAB 7.12.0 (R2011a), and it was evaluated using color images. The test images used in the experiments include the three famous benchmark images: *Lena*, *Baboon* and *Peppers*. Quality of the reconstructed images was determined by measuring the PSNR.

Figure 31: Pepper image

Figure 30: Lena image

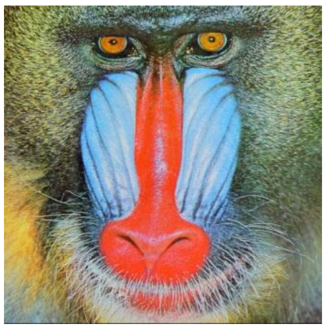


Figure 29: Baboon image

In our application, there are several parameters affect the obtained results such as the logarithmic used function, the exponentiation base for the quantization step, and shifting value. For reason to achieve good results, a set of experiments has performed with different values of parameters.

Thus, to organize this section, we present these experiments and its results in sequence which is based on the chosen parameters.

2.1 Experiment one

In this first experiment, we set the following values of input parameters:

- Logarithm transform based on $\log_2$
- Shifting value equal to 256

Where the exponent bases for quantization phase are varying from 2 to 8, the Table 11 shows the obtained results.

Table 11: Obtained results for the first experiment

Exponent base	Lena		Pepper		Baboon	
	CR	PSNR	CR	PSNR	CR	PSNR
2	14.1768	29.2036	12.9584	29.1013	6.5745	29.3722
3	8.8434	35.2858	7.4369	34.5767	4.2432	35.2344
4	6.5234	38.5689	5.6081	37.5509	3.3950	38.5618
5	5.3749	40.7597	4.6882	39.5442	2.9468	40.7760
6	4.6739	42.4589	4.1047	40.7288	2.6657	42.4470
7	4.1829	43.6722	3.7169	41.9387	2.4699	43.6963
8	3.8520	44.7673	3.4300	42.7428	2.3274	44.7589

2.2 Experiment two

In this second experiment, we set the following values of input parameters:

- Logarithm transform based on $\log_2$
- Shifting value equal to 512

Where the exponent bases for quantization phase are varying from 2 to 18, the Table 12 shows the obtained results.

Table 12: Obtained results for the second experiment

Exponent base	Lena		Pepper		Baboon	
	CR	PSNR	CR	PSNR	CR	PSNR
2	26.0930	23.4844	23.9991	23.2066	10.5830	23.2367
3	16.4163	28.5239	13.7667	27.6684	7.1474	28.4352
4	12.4550	31.1957	10.9190	30.5084	5.7104	31.1393

5	10.4524	32.9963	9.3453	32.2876	4.9754	33.0118
6	9.1344	34.2965	8.0227	33.4240	4.4862	34.3765
10	7.1406	37.2188	6.1745	36.9159	3.6596	37.1978
18	5.3379	40.4740	4.6627	39.8971	2.9444	40.4458

2.3 Experiment three

In this third experiment, we set the following values of input parameters:

- Logarithm transform based on $\log_3$
- Shifting value equal to 244

Where the exponent bases for quantization phase are varying from 2 to 17, the Table 13 shows the obtained results.

Table 13: Obtained results for the third experiment

Exponent base	Lena		Pepper		Baboon	
	CR	PSNR	CR	PSNR	CR	PSNR
2	24.1221	24.6594	19.1721	23.9478	10.0246	24.6474
4	10.7642	32.6584	9.2984	32.2731	5.0926	32.7081
5	9.2655	34.7811	7.7828	34.1375	4.4082	34.6844
6	7.9617	36.1548	6.9033	35.5545	4.0006	36.0724
8	6.6584	38.3618	5.6883	37.5288	3.4439	38.3161
12	5.2733	41.0066	4.6017	39.9969	2.9061	40.9863
17	4.4496	43.0376	3.9038	41.8830	2.5724	43.0811

2.4 Experiment four

In this fourth experiment, we set the following values of input parameters:

- Logarithm transform based on $\log_3$
- Shifting value equal to 729

Where the exponent bases for quantization phase are varying from 2 to 255, the Table 14 shows the obtained results.

Table 14: Obtained results for the fourth experiment

Exponent base	Lena		Pepper		Baboon	
	CR	PSNR	CR	PSNR	CR	PSNR
2	74.5566	16.8712	68.1664	16.3978	23.6848	16.4429
7	20.3928	26.4735	17.7672	25.6436	8.6053	26.3850
12	15.4950	29.1576	13.4629	28.2912	6.7331	28.9510
24	11.5392	32.0239	10.2579	31.1621	5.3734	31.9650
36	10.1112	33.4737	8.9525	32.5638	4.8012	33.3967
64	8.3610	35.2596	7.3033	34.1464	4.1746	35.3279
255	5.9501	39.2339	5.2087	37.6260	3.1804	39.1890

2.5 Experiment five

In this fifth experiment, we set the following values of input parameters:

- Logarithm transform based on $\log_{10}$
- Shifting value equal to 100

Where the exponent bases for quantization phase are varying from 2 to 26, the Table 15 shows the obtained results.

Table 15: Obtained results for the fifth experiment

Exponent base	Lena		Pepper		Baboon	
	CR	PSNR	CR	PSNR	CR	PSNR
2	30.6835	22.7501	23.3480	22.3734	11.6333	22.1711
3	16.7909	27.3275	14.6573	27.4155	7.6536	27.6017
7	9.0459	35.0115	7.4809	34.8422	4.3164	35.0071
10	7.2303	37.4777	6.0668	37.1120	3.6522	37.4681
14	6.0407	39.6741	5.1064	39.1608	3.1921	39.6643
18	5.3519	41.1233	4.5811	40.3958	2.9269	41.1191
26	4.5524	43.1286	3.9389	42.3471	2.6106	43.1464

2.6 Experiment six

In this sixth experiment, we set the following values of input parameters:

- Logarithm transform based on $\log_{10}$
- Shifting value equal to 1000

Where the exponent bases for quantization phase are varying from 2 to 255, the Table 16 shows the obtained results.

Table 16: Obtained results for the sixth experiment

Exponent base	Lena		Pepper		Baboon	
	CR	PSNR	CR	PSNR	CR	PSNR
2	2313.4	8.1294	161.1828	7.2033	1937.4	8.1476
4	119.57	14.9303	57.7894	12.9936	83.1471	14.4781
20	34.2494	20.1145	29.3716	20.5351	14.0537	20.1054
32	31.5994	21.4546	28.4969	22.4203	12.4387	21.6486
64	25.0079	23.5190	22.1563	23.8825	10.5093	23.3573
127	23.8652	25.4815	17.9784	25.1043	9.4544	25.1407
255	19.9057	26.7982	15.8516	26.3833	8.4430	26.6879

3. Examples of decompressed images

Appendix B shows some decompressed images as results of the use of different parameters such as shifted values and exponential base for quantization.

4. Comparative Study Results

The proposed method has been compared with two well-known image compression standards –JPEG and JPEG 2000 for three benchmark image: *Lena, Pepper, and Baboon.*

Compared with JPEG standard, our method has gained better compression for the PSNR above 35 db.

Compared with JPEG 2000 standard, our method has gained better compression for the PSNR above 40 db.

Where for the benchmark image *Baboon*, our method has gained better compression ratios compared with both JPEG and JPEG 2000 for PSNR above 31db.

Figure 32 and 33; illustrate such a comparative study for the three images.

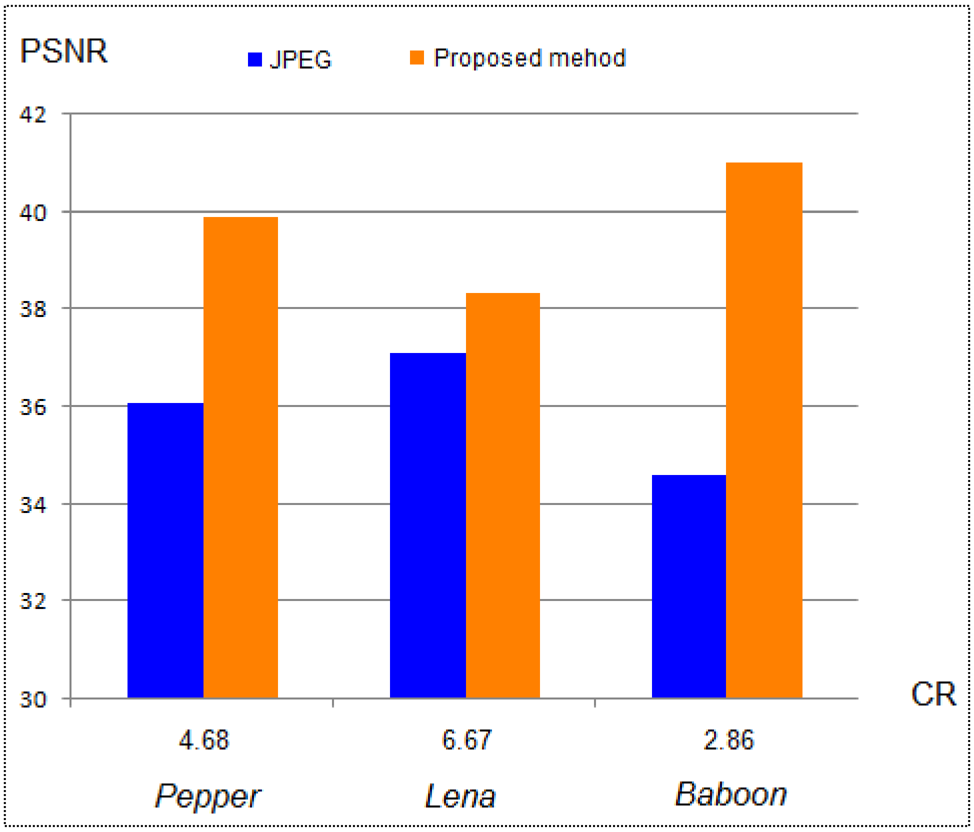


Figure 32: Comparison between JPEG and Proposed method

The result of JPEG is obtained using **FILEminimizer Pictures** software[1].

The result of JPEG 2000 is obtained using MATLAB 7.12.0 (R2011a) JPEG2000 implementation.

[1] FILEminimizer Pictures offers users to reduce the weight of format image files JPEG, GIF, TIFF, PNG, BMP, or EMF. Image compression algorithms allow up to 98% space savings while preserving the quality of the final image. All features are available in one click. An optimization option to send as attachments via email is also part of the game. Downloaded from: http://fileminimizer-pictures.en.softonic.com/download.

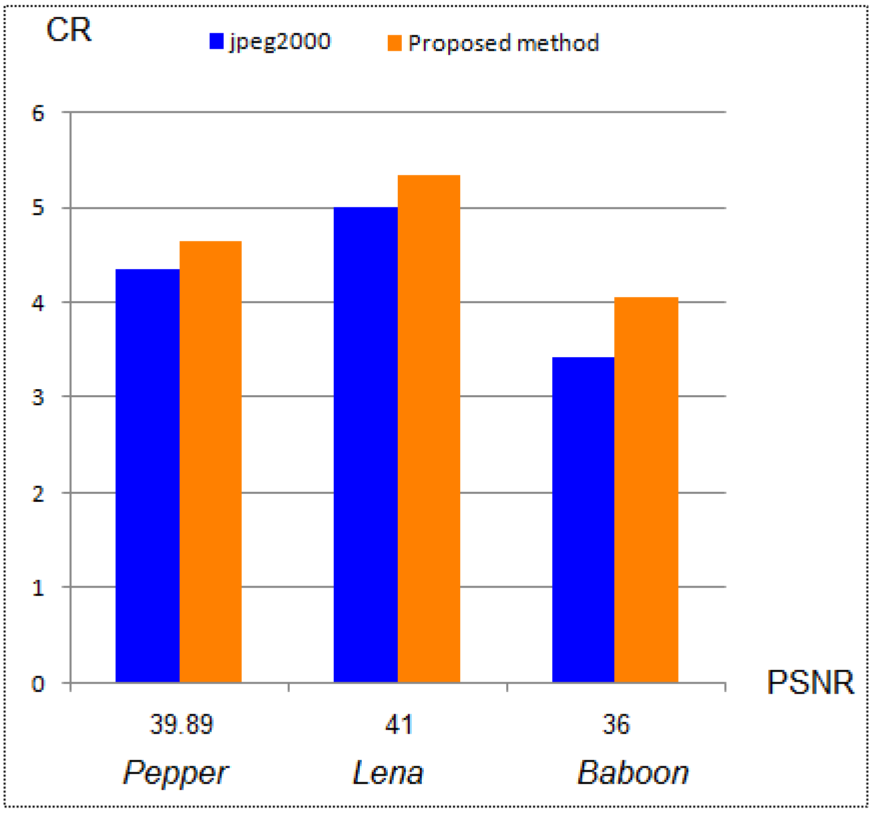


Figure 33: Comparison between JPEG2000 and Proposed method

5. Performance Evaluation

In the previous section, we present our proposed method which provides sufficient high compression ratios with no appreciable degradation of image quality.

The effectiveness and robustness of our approach has been justified by the experimental results which have been arranged in tables 1, 2…6; and to demonstrate the performance of the proposed method, a comparison between the proposed technique and other common compression techniques has been performed.

The principal advantages of our method are the flexibility and the processing time reduction[1] which is proportional with the entropy encoding used technique.

The evaluation of our proposed method can be performed by considering the three following measurements:

[1] Compared with tow related works discussed in chapter 3 which were used the same transformation based on logarithm.

5.1 Compression ratio

Our approach can get high compression ratio for the complex texture image (e.g. benchmark image Baboon), especially for the PSNR above 40 db.

The compression ratio depends directly on the exponent base for the quantization phase, and like the compression methods which are based on transformation in first stage, in our method, the compression ratio depends on the image contents (entropy); thereby, we can notice the difference in compression ratios for different images at the same quantization value.

5.2 Quality of the reconstructed image

For high compression ratio, our proposed method can achieve high quality of the decompressed image, while the quality is exchanging in contrast with the compression ratio.

Because our method is a lossy compression technique, it causes degradations of the reconstructed image in exchange for larger compression ratios. Due, the pixel-by-pixel processing style is useful to avoid the block artifact for the high quality image compression (Huang, Chen, & Chang, 1999).

5.3 Computing Time

Computing time plays a major role in selecting a compression scheme (Kurt & Fayez, June 1996). Computing time in our method is independent of the quantization factor and the compression ratio. In fact, it depends only of the entropy encoding used scheme.

The time required to compress a 512 by 512 color image using our proposed method in average is about 6.5 second. Where, the time required of decompression in average is about 6.0 second.

CONCLUSION

In this chapter, we have introduced a novel image compression method based on logarithm transformation which provides sufficient high compression ratios with no appreciable degradation of image quality. The performance of our proposed method has been evaluated using a set of benchmark images in a sequence of experiences. In order to demonstrate the performance of the proposed method, a comparison between the proposed technique and other common compression techniques has been performed.

From the experimental results it is clearly shown that the proposed compression technique gives better performance compared to other traditional techniques. Hence, logarithm-based transform is a better suited scheme for image compression.

CONCLUSION

In recent years, image compression has wide area in image processing researches. This importance is resulted by the increasing development of information technology in our world, where the image is a substantial key in several domains as press, publicities, satellites communication, geographic, etc. People all over the world are sharing, transmitting and storing millions of images every moment. This wide use of image makes production of digital images is being increased too in that proportion. Consequently, the demand of effective image compression algorithms is yet very high.

Keeping this demand into mind, a new image compression method has been introduced in this research which entitled Logarithm Transform Based Image Compression with Pixels Shifting. First, an image in question to compress is shifted at the pixel level. Then, the transformation based on logarithmic functions is applied. Next, in the quantization stage which can be considered the heart of lossy algorithm scheme, we have proposed a non-uniform quantization in order to perform compression. Finally, to benefit better of the entropy encoding which is totally lossless, we have applied an efficient schemes.

Based on our experiment results, our proposed method gives better compression results compared with other methods, which proves the above statement. Also, it shows that application of logarithm transform in the transformation phase gives a slightly improved performance on average compared to the JPEG 2000 standard, especially for PSNR above 40 (dB). Also, a significant improved performance on average is produced compared to the JPEG standard, especially for PSNR above 35 (dB).

The effect of using different exponential bases played an important role while processing a non-uniform quantization by compression. The experimental result shows that the reconstructed image compression with higher bases is better than smaller base.

For entropy coding, the uses of the **paq8pxd_v4** or other variants of the **paq** family can yield better average results than applying only one of the common lossless software such as **WinRAR**, etc. The storage as bitmap format after compressing using paq8pxd_v4 gives better compression ratios compared to other format.

Our proposed method obtains better compression results. However, the run time increases significantly to achieve a high compression ratio because the use of **paq8pxd_v4**. Also, the proposed method has been tested and evaluated only for three benchmark images which are Lena, Pepper, and Baboon. In order to gain high confidence, our work needs to be tested for other benchmark images.

As an application, the proposed method gives the user an easier way to do a variety of tests about logarithmic functions transform for image compressions. It could be improved in several ways:

- Exploit more properties of logarithmic functions.
- Combined with other reductive functions as Sine, Cosine…
- Hybridized with other transform techniques as DST, DCT, and DWT…
- Change the quantization type using other technique.
- Apply logarithm transformation by block or region like fractal technique.

APPENDIX A

Here, we give an example of Baseline compression and encoding of a single 8x8 sample block. We chose in this example a logarithmic function based on $\log_2$ for forward transformation, and a value equal to 256 for pixels shifting, and 10 as base for the quantization. Note that is not a complete process for the proposed method. Nonetheless, this example should help to make concrete much of the foregoing explanation.

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

(a) source image samples

395	400	405	409	411	411	411	411
400	407	409	412	415	412	412	412
406	411	416	419	414	412	412	412
415	417	418	416	416	415	415	415
415	416	417	418	418	411	411	411
417	417	417	417	416	413	413	413
418	418	417	419	418	413	413	413
418	418	417	417	419	414	414	414

(b) pixels shifting by adding 256

8.6257	8.6439	8.6618	8.6760	8.6830	8.6830	8.6830	8.6830
8.6439	8.6689	8.6760	8.6865	8.6970	8.6865	8.6865	8.6865
8.6653	8.6830	8.7004	8.7108	8.6935	8.6865	8.6865	8.6865
8.6970	8.7039	8.7074	8.7004	8.7004	8.6970	8.6970	8.6970
8.6970	8.7004	8.7039	8.7074	8.7074	8.6830	8.6830	8.6830
8.7039	8.7039	8.7039	8.7039	8.7004	8.6900	8.6900	8.6900
8.7074	8.7074	8.7039	8.7108	8.7074	8.6900	8.6900	8.6900
8.7074	8.7074	8.7039	8.7039	8.7108	8.6935	8.6935	8.6935

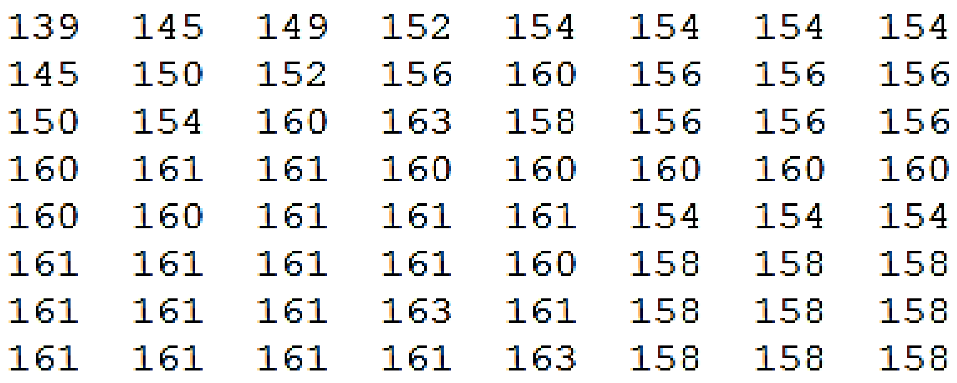
(c) Logarithm transform based on $\log_2$

0.6257	0.6439	0.6618	0.6760	0.6830	0.6830	0.6830	0.6830
0.6439	0.6689	0.6760	0.6865	0.6970	0.6865	0.6865	0.6865
0.6653	0.6830	0.7004	0.7108	0.6935	0.6865	0.6865	0.6865
0.6970	0.7039	0.7074	0.7004	0.7004	0.6970	0.6970	0.6970
0.6970	0.7004	0.7039	0.7074	0.7074	0.6830	0.6830	0.6830
0.7039	0.7039	0.7039	0.7039	0.7004	0.6900	0.6900	0.6900
0.7074	0.7074	0.7039	0.7108	0.7074	0.6900	0.6900	0.6900
0.7074	0.7074	0.7039	0.7039	0.7108	0.6935	0.6935	0.6935

(d) The integer part elimination

1.5430	1.5625	1.5820	1.5977	1.6055	1.6055	1.6055	1.6055
1.5625	1.5898	1.5977	1.6094	1.6211	1.6094	1.6094	1.6094
1.5859	1.6055	1.6250	1.6367	1.6172	1.6094	1.6094	1.6094
1.6211	1.6289	1.6328	1.6250	1.6250	1.6211	1.6211	1.6211
1.6211	1.6250	1.6289	1.6328	1.6328	1.6055	1.6055	1.6055
1.6289	1.6289	1.6289	1.6289	1.6250	1.6133	1.6133	1.6133
1.6328	1.6328	1.6289	1.6367	1.6328	1.6133	1.6133	1.6133
1.6328	1.6328	1.6289	1.6289	1.6367	1.6172	1.6172	1.6172

(e) Quantization



139	145	149	152	154	154	154	154
145	150	152	156	160	156	156	156
150	154	160	163	158	156	156	156
160	161	161	160	160	160	160	160
160	160	161	161	161	154	154	154
161	161	161	161	160	158	158	158
161	161	161	163	161	158	158	158
161	161	161	161	163	158	158	158

(g) Reconstructed image samples

APPENDIX B

These three images are obtained, with compression of Pepper when shiftValue parameter is set into 256 and logBase parameter is set into 2:

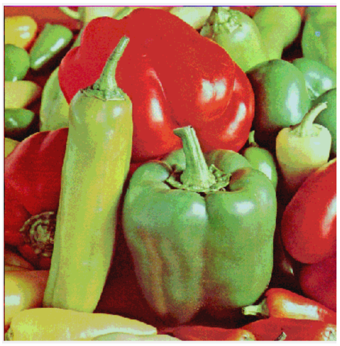


Figure 35: exponential base parameter set into 2 PSNR: 29.10

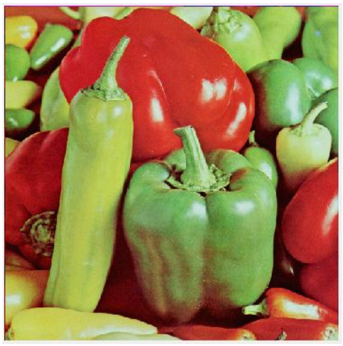


Figure 36: exponential base parameter set into 5 PSNR: 39.54

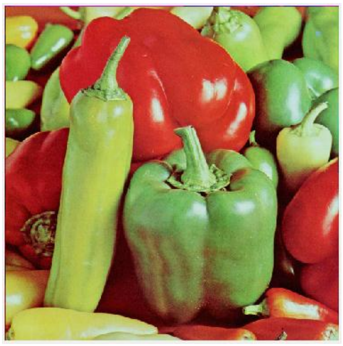


Figure 34: exponential base parameter set into 8 PSNR: 42.74

These other images are obtained with compression of Pepper when shiftValue parameter is set into 512 and logBase parameter is set into 2:

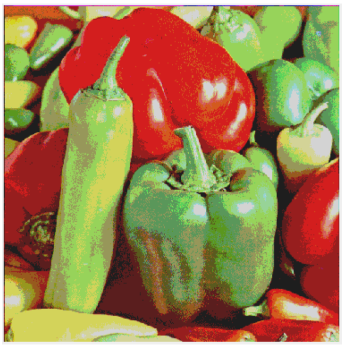


Figure 38: exponential base parameter set into 2 PSNR: 23.20

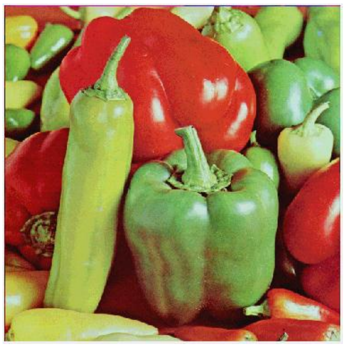


Figure 37: exponential base parameter set into 5 PSNR: 32.28

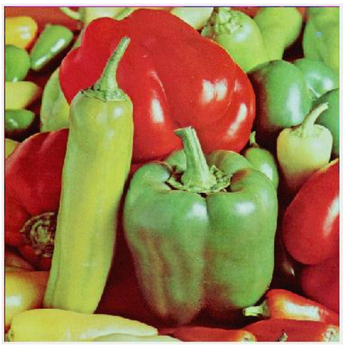


Figure 39: exponential base parameter set into 8 PSNR: 35.12

And shiftValue is set into 1024 and logBase is set into 2:

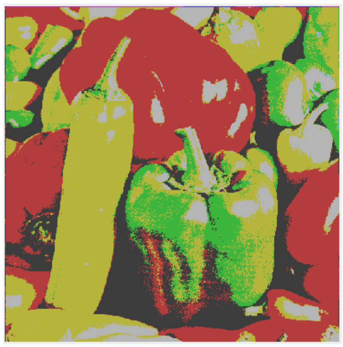


Figure 42: exponential base parameter set into 2 PSNR: 18.14

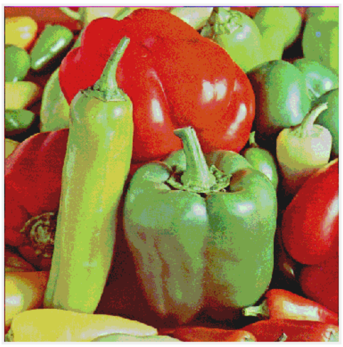


Figure 41: exponential base parameter set into 5 PSNR: 25.36

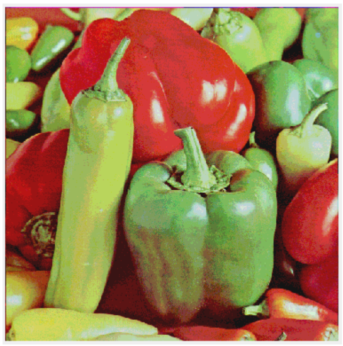


Figure 40: exponential base parameter set into 8 PSNR: 27.80

BIBLIOGRAPHY

Adler.et.al. (1996). *PNG (PORTABLE NETWORK GRAPHICS) SPECIFICATION* (éd. Version 1.0). (T. Boutell, Éd.) Massachusetts Institute of Technology (MIT).

AMEY, J. C. (2001-2002). *co dage des cou leurs* . TPE.

Barnsley, & et.al. (1988). Harnessing chaos for image synthesis. *Computer Graphics, vol 22*(4), 131-140.

Bell, T., Witten, I., & Cleary, J. (1989). Modeling for Text Compression. *ACM Computing Surveys, Vol. 21*(4).

Bernstein, S. a. (1999). *Schaum's outline of theory and problems of elements of statistics. I, Descriptive statistics and probability, Schaum's outline series.* New York: McGraw-Hill.

Blelloch, E. (2002). *Introduction to Data Compression.* Computer Science Department, Carnegie Mellon University.

Boynton, P. a. (1996). Human Color Vision. Washington: Optical Society ofAmerica.

Cavet. (1992). Le système Photo CD de Kodak. *Note technique CNDP*.

Cohen, A., & Resnikoff, H. (1994). Image Compression for Radiology and Telemedicine. *Proc.SPIE 2298*, 304-315.

Curry, D., Matinsen, G., & Hopper, D. (2003). Capability of the human visual. *Air Force Research Laboratory, Cockpit Displays X, Darrel Hopper, 5080*, 58-69.

Few, S. (February 2008). Practical Rules for Using Color in Charts. *Perceptual Edge* (pp. 1-13). Visual Business Intelligence Newsletter.

Firas A. Jassim, H. E. (October 2012). FIVE MODULUS METHOD FOR IMAGE. *Signal & Image Processing : An International Journal (SIPIJ), Vol.3*(No.5), 02.

Frey, B. (2006). *Statistics hacks.* Hacks Series, Sebastopol, CA: O'Reilly.

G.Strang. (1999). *The Discrete Cosine Transform.* Society for Industrial and Applied Mathematics.

Gaurav, V., Sanjay, S., & Rajeev, P. (October 2013). A Survey: Various Techniques of Image Compression. *International Journal of Computer Science and Information Security, Vol. 11*(10).

Gu.H. (2000). *Image Compression Using the Haar Wavelet Transform.* Masters thesis, East Tennessee State University.

Han, Y., & Leou, J. (April 1998). Detection and Correction of Transmission Errors in JPEG Images. *IEEE Trans. Circuits and Systems for Video Technology*, 221-231.

Huang, S., & Chen, L. (1998). LOG-EXP Compression System Design and Implementation. *IEEE International Sym. on Consumer Electronics,.*

Huang, S.-C., Chen, L.-G., & Chang, H.-C. (1999). A Novel Image Compression Algorithm by Using LOG-EXP Transform. *IEEE Sym. Circuits and Systems*, 17-20.

HUFFMAN, D. (1952, September). A method for the construction of minimum redundancy codes. *40*(9), 1098-1101.

Huffman, K. (September 1991). *Profile: David A. Huffman.* Scientific American.

Ian , Y. T., Jan , G. J., & Lucas , v. J. (2007). *Fundamentals of Image Processing* (éd. Version 2.3). Delft University of Technology.

Kate, & Bhapkar. (2009). *Basics Of Mathematics.* Pune: Technical Publications,.

KODITUWAKKU, & AMARASINGHE. (2015). COMPARISON OF LOSSLESS DATA COMPRESSION ALGORITHMS FOR TEXT DATA. *Indian Journal of Computer Science and Engineering, Vol 1*(4), 416-425.

Kurt, N., & Fayez, S. (June 1996). A Comparison of Two lmage Compression Techniques for Softcopy Photogrammetry. *Photogrammetric Engineering & Remote Sensing, Vol. 62*(6), 695-701.

M.Irani, P. A. (1996). Efficient representations of vidéo sequences and their applications. *Signal Processing : Image Communication*(8), 327-351.

Mozammel, H. ,., & Amina, K. (2012, July). Image Compression Using Discrete Wavelet Transform. *International Journal of Computer Science Issues, Vol. 9*(4).

Mridul, K. ,., Seema, L., & Dheeraj, S. (2012). Lossless Huffman Coding Technique For Image Compression And Reconstruction Using Binary Trees. *IJCTA*, 76-79.

Natarajan, S., & Ramana, R. (September 2008). Modified LOG-EXP Based Image Compression Algorithm. *International Journal of Computer Science and Network Security*.

Ohio, C. a. (1990). *GRAPHICS INTERCHANGE FORMAT(sm)* (éd. Version 89a). CompuServe Incorporated.

Paquel. (1993). Multimédia et télévision interactive. *Mémoires optiques et systèmes*.

PLUME, P. (1993). *Compression de données.* Editions Eyrolles.

Rabbani, M., & Jones, P. W. (1991). Digital Image-Compression Techniques. (s. Editor, Éd.) *D.C , O'S hea , vol. TT7*.

Rfael, Y., & Richard, W. (2002). *Digital Image Processing* (éd. 2nd edition). Prentice Hall.

Ricciardi, L. M. (1990). *Lectures in applied mathematics and informatics.* Manchester: Manchester University Press.

RISSANEN, & LANGDON, A. (1979). Arithmetic coding. *IBM J. Res. Dev*, 1491-62.

Rudin, W. (1987). *Real and complex analysis* (éd. 3rd ed). New York: McGraw-Hill.

Salles, D. (mars 2005). *Éducation à l'image et aux médias : la liberté de la presse.* Centre de Ressources en éducation aux médias CREM.

Salomon, D. (1998). *Data Compression, The Complete Reference* (éd. 2nd Edition). Springer-Verlag.

Sayood, K. (2000). *Introduction to Data Compression* (éd. Second edition). Academic Press.

Sethy, R. R., & Swastik, D. (2009). *DIGITAL IMAGE COMPRESSION USING DISCRETE COSINE TRANSFORM & DISCRETE WAVELET TRANSFORM.* Rourkela-769 008,Orissa, India: Department of Computer Science andEngineering National Institute of Technology Rourkela.

Shirali, S. (2002). *A Primer on Logarithms.* Hyderabad: Universities Press,.

Sonal, D. K. (2015). *A STUDY OF VARIOUS IMAGE COMPRESSION TECHNIQUES.* Hisar: Guru Jhambheswar University of Science and Technology.

THON, S. (2013-2014). *Imagerie Numérique Représentation et codage des images.* Marseille: Institut Universitaire de technologie.

WAGNER, C. (1993). *De l'image vers la compression.* Rapport de Recherch e de l' INRIA.

Wallace, G. K., & et.al. (1991). The JPEG Still Picture Compression Standard. *IEEE Transactions on Consumer Electronics*.

Wei, W.-Y. (s.d.). *An Introduction to Image Compression.* Graduate Institute of Communication Engineering. Taiwan: Graduate Institute of Communication Engineering National Taiwan University, Taipei , Taiwan, ROC.

Whitrow, R. (2008). *Image File Formats*. (S.-V. L. Limited, Éd.) London : OpenGL Graphics Through Applications,.

Wolfram, S. (2002). A New Kind of Science. *Champaign, IL: Wolfram Media*(1069).

Yusra, A.-N., & Chen, S. D. (2012, August). Comparison of Image Quality Assessment: PSNR, HVS, SSIM, UIQI. *International Journal of Scientific & Engineering Research, Volume 3*(8), 02.

Ziv, & Lempel. (1977). A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory, Vol. 23*(3), 337343.

Ziv, & Lempel. (1978). Compression ofIndividual Sequences via VariableRate Coding. *IEEE Transactions on Information Theory, Vol. 24*(5), 530-536.