INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN: 2278 – 0211 (Online)

Design, Testing And Implementation Of Digital Image Processing Systems For Viable Solutions

Dr. G. Manoj Someswar
Principal & Professor, Department Of Computer Science & Engineering
Anwar-Ul-Uloom College Of Engineering & Technology, Yennepally, RR District, Vikarabad, A.P., India
V. Krishna Naik
Assistant Professor, Department Of Electronics & Communications Engineering
Axsum University, Ethiopia, Africa
Dayananda R. B.
Assistant Professor, Department Of Computer Science & Engineering
Raja Reddy Institute Of Technology, Chikkabanavara, Bangalore, Karnataka, India

Abstract:
Digital image processing is an area characterized by the need for extensive experimental work to establish the viability of proposed solutions to a given problem. An important characteristic underlying the design of image processing systems is the significant level of testing & experimentation that normally is required before arriving at an acceptable solution. This characteristic implies that the ability to formulate approaches & quickly prototype candidate solutions generally plays a major role in reducing the cost & time required to arrive at a viable system implementation.
An image may be defined as a two-dimensional function f (x, y), where x & y are spatial coordinates, & the amplitude off at any pair of coordinates (x, y) is called the intensity or gray level of the image at that point. When x, y & the amplitude values of f are all finite discrete quantities, we call the image a digital image. The field of DIP refers to processing digital image by means of digital computer. The digital image is composed of a finite number of elements, each of which has a particular location & value. The elements are called pixels.
Vision is the most advanced of our sensor, so it is not surprising that image plays the single most important role in human perception. However, unlike humans, who are limited to the visual band of the EM spectrum imaging machines cover almost the entire EM spectrum, ranging from gamma to radio waves. They can operate also on images generated by sources that humans are not accustomed to associating with image.
There is no general agreement among authors regarding where image processing stops & other related areas such as image analysis& computer vision start. Sometimes a distinction is made by defining image processing as a discipline in which both the input & output of a process are images. This is limiting & somewhat artificial boundary. The area of image analysis (image understanding) is in between image processing & computer vision.
There are no clear-cut boundaries in the continuum from image processing at one end to complete vision at the other. However, one useful paradigm is to consider three types of computerized processes in this continuum: low-, mid-, & high-level processes. Low-level process involves primitive operations such as image processing to reduce noise, contrast enhancement & image sharpening. A low- level process is characterized by the fact that both its inputs & outputs are images. Mid-level process on images involves tasks such as segmentation, a description of that object to reduce them to a form suitable for computer processing & classification of individual objects. A mid-level process is characterized by the fact that its inputs generally are images but its outputs are attributes extracted from those images. Finally higher- level processing involves "Making sense" of an ensemble of recognized objects, as in image analysis & at the far end of the continuum performing the cognitive functions normally associated with human vision.
Digital image processing, as already defined is used successfully in a broad range of areas of exceptional social & economic value.

***Key words**: Image analysis, Matrices, Data classes, IMAQ vision, LabView*

1.Introduction

An image is represented as a two dimensional function f(x, y) where x and y are spatial co-ordinates and the amplitude of 'f' at any pair of coordinates (x, y) is called the intensity of the image at that point.

1.1.Gray Scale Image

A grayscale image is a function I (xylem) of the two spatial coordinates of the image plane.

I(x, y) is the intensity of the image at the point (x, y) on the image plane.

I (xylem) takes non-negative values assume the image is bounded by a rectangle [0, a] ×[0, b]I: [0, a] × [0, b] → [0, info)

1.2.Colour Image

It can be represented by three functions, R (xylem) for red, G (xylem) for green and B (xylem) for blue. [6]

An image may be continuous with respect to the x and y coordinates and also in amplitude. Converting such an image to digital form requires that the coordinates as well as the amplitude to be digitized. Digitizing the coordinate's values is called sampling. Digitizing the amplitude values is called quantization.

1.3.Coordinate Convention

The result of sampling and quantization is a matrix of real numbers. We use two principal ways to represent digital images. Assume that an image f(x, y) is sampled so that the resulting image has M rows and N columns. We say that the image is of size M X N. The values of the coordinates (xylem) are discrete quantities. For notational clarity and convenience, we use integer values for these discrete coordinates. In many image processing books, the image origin is defined to be at (xylem)=(0,0).The next coordinate values along the first row of the image are (xylem)=(0,1).It is important to keep in mind that the notation (0,1) is used to signify the second sample along the first row. It does not mean that these are the actual values of physical coordinates when the image was sampled. Following figure shows the coordinate convention. Note that x ranges from 0 to M-1 and y from 0 to N-1 in integer increments.[7]The coordinate convention used in the toolbox to denote arrays is different from the preceding paragraph in two minor ways. First, instead of using (xylem) the toolbox uses the notation (race) to indicate rows and columns. Note, however, that the order of coordinates is the same as the order discussed in the previous paragraph, in the sense that the first element of a coordinate topples, (alb), refers to a row and the second to a column. The other difference is that the origin of the coordinate system is at (r, c) = (1, 1); thus, r ranges from 1 to M and c from 1 to N in integer increments. IPT documentation refers to the coordinates. Less frequently the toolbox also employs another coordinate convention called spatial coordinates which uses x to refer to the columns and y to refer to rows. This is the opposite of our use of variables x and y. [1]

1.4.Image As Matrices

The preceding discussion leads to the following representation for a digitized image function:

f (0,0) f(0,1) …….. f(0,N-1) f(1,0) f(1,1) …… f(1,N-1)

f(xylem)= . . f(M-1,0) f(M-1,1) …… f(M-1,N-1)

The right side of this equation is a digital image by definition. Each element of this array is called an image element, picture element, pixel or pel. The terms image and pixel are used throughout the rest of our discussions to denote a digital image and its elements.[2]

A digital image can be represented naturally as a MATLAB matrix:

f(1,1) f(1,2) ……. f(1,N)

f(2,1) f(2,2) …….. f(2,N)

f = f(M,1) f(M,2) …….f(M,N)

Where f(1,1) = f(0,0) (note the use of a monoscope font to denote MATLAB quantities). Clearly the two representations are identical, except for the shift in origin. The notation f(p ,q) denotes the element located in row p and the column q. For example f(6,2) is the element in the sixth row and second column of the matrix f. Typically we use the letters M and N respectively to denote the number of rows and columns in a matrix. A 1xN matrix is called a row vector whereas an Mx1 matrix is called a column vector. A 1x1 matrix is a scalar.[5]

Matrices in MATLAB are stored in variables with names such as A, a, RGB, real array and so on. Variables must begin with a letter and contain only letters, numerals and underscores. As noted in the previous paragraph, all MATLAB quantities are written using mono-scope characters. We use conventional Roman, italic notation such as f(x ,y), for mathematical expressions.[3]

1.5.Reading Images

Images are read into the MATLAB environment using function imread whose syntax is imread('filename').

Format Name	Description	Recognized Extension
TIFF	Tagged Image File Format	.tif, .tiff
JPEG	Joint Photograph Experts Group	.jpg, .jpeg
GIF	Graphics Interchange Format	.gif
BMP	Windows, Bitmap	.bmp
PNG	Portable Network Graphics	.png
XWD	X Window Dump	.xwd

Table 1

Here, the filename is a spring containing the complete of the image file (including any applicable extension). For example the command line

>> f = imread (‘8. jpg’);

reads the JPEG (above table) image chestxray into image array f. Note the use of single quotes (‘) to delimit the string filename. The semicolon at the end of a command line is used by MATLAB for suppressing output. If a semicolon is not included. MATLAB displays the results of the operation(s) specified in that line. The prompt symbol(>>) designates the beginning of a command line, as it appears in the MATLAB command window.[4]

When as in the preceding command line no path is included in filename, imread reads the file from the current directory and if that fails it tries to find the file in the MATLAB search path. The simplest way to read an image from a specified directory is to include a full or relative path to that directory in filename.

For example,

>> f = imread (‘D:\myimages\chestxray.jpg’);

reads the image from a folder called my images on the D: drive, whereas >> f = imread(‘ . \ myimages\chestxray .jpg’); reads the image from the my images subdirectory of the current of the current working directory. The current directory window on the MATLAB desktop toolbar displays MATLAB’s current working directory and provides a simple, manual way to change it. Above table lists some of the most of the popular image/graphics formats supported by imread and imwrite.[4]

Function size gives the row and column dimensions of an image:

>> size (f)

ans = 1024 * 1024

This function is particularly useful in programming when used in the following form to determine automatically the size of an image:

>>[M,N]=size(f);

This syntax returns the number of rows (M) and columns (N) in the image. [5]

The whole function displays additional information about an array. For instance ,the statement >> whos f gives

Name	size	Bytes	Class
F	1024*1024	1048576	unit8 array

Table 2

The grand total is 1048576 elements using 1048576 bytes. The unit8 entry shown refers to one of several MATLAB data classes. A semicolon at the end of a whose line has no effect , so normally one is not used.

2.Experimentation Design & Setup

Images are displayed on the MATLAB desktop using function imshow, which has the basic syntax:

imshow(f,g)

Where f is an image array, and g is the number of intensity levels used to display it. If g is omitted ,it defaults to 256 levels .using the syntax

imshow(f,{low high})

Displays as black all values less than or equal to low and as white all values greater than or equal to high. The values in between are displayed as intermediate intensity values using the default number of levels .Finally the syntax

Imshow(f,[

Sets variable low to the minimum value of array f and high to its maximum value. This form of imshow is useful for displaying images that have a low dynamic range or that have positive and negative values.

Function pixval is used frequently to display the intensity values of individual pixels interactively. This function displays a cursor overlaid on an image. As the cursor is moved over the image with the mouse the coordinates of the cursor position and the corresponding intensity values are shown on a display that appears below the figure window .When working with colour images, the

coordinates as well as the red, green and blue components are displayed. If the left button on the mouse is clicked and then held pressed, pixval displays the Euclidean distance between the initial and current cursor locations.
The syntax form of interest here is Pixval which shows the cursor on the last image displayed. Clicking the X button on the cursor window turns it off.
The following statements read from disk an image called rose_512.tif extract basic information about the image and display it using imshow :
>>f=imread('rose_512.tif');
>>whos f

Name	Size	Bytes	Class
F	512*512	262144	unit8 array

Table 3

Grand total is 262144 elements using 262144 bytes
>>imshow(f)
A semicolon at the end of an imshow line has no effect, so normally one is not used. If another image, g, is displayed using imshow, MATLAB replaces the image in the screen with the new image. To keep the first image and output a second image, we use function figure as follows:
>>figure ,imshow(g)
Using the statement
>>imshow(f),figure ,imshow(g) displays both images.
Note that more than one command can be written on a line ,as long as different commands are properly delimited by commas or semicolons. As mentioned earlier, a semicolon is used whenever it is desired to suppress screen outputs from a command line.
Suppose that we have just read an image h and find that using imshow produces the image. It is clear that this image has a low dynamic range, which can be remedied for display purposes by using the statement.
>>imshow(h,[ ])

3.Writing Images
Images are written to disk using function imwrite, which has the following basic syntax:
Imwrite (f,'filename')
With this syntax, the string contained in filename must include a recognized file format extension .Alternatively, the desired format can be specified explicitly with a third input argument. >>imwrite(f,'patient10_run1','tif')
Or alternatively
For example the following command writes f to a TIFF file named patient10_run1:
>>imwrite(f,'patient10_run1.tif')
If filename contains no path information, then imwrite saves the file in the current working directory.
The imwrite function can have other parameters depending on e file format selected. Most of the work in the following deals either with JPEG or TIFF images ,so we focus attention here on these two formats.
More general imwrite syntax applicable only to JPEG images is
imwrite(f,'filename.jpg,,'quality',q)
Where q is an integer between 0 and 100 (the lower the number the higher the degradation due to JPEG compression).
For example, for q=25 the applicable syntax is
>> imwrite(f,'bubbles25.jpg','quality',25)
The image for q=15 has false contouring that is barely visible, but this effect becomes quite pronounced for q=5 and q=0.Thus, an expectable solution with some margin for error is to compress the images with q=25.In order to get an idea of the compression achieved and to obtain other image file details, we can use function imfinfo which has syntax.
Imfinfo filename
Here filename is the complete file name of the image stored in disk.
For example,
>> imfinfo bubbles25.jpg
outputs the following information(note that some fields contain no information in this case):

Filename: 'bubbles25.jpg'
FileModDate: '04-Jan-2003 12:31:26'
FileSize: 13849
Format: 'jpg'
Format Version: ' '
Width: 714

Height: 682
Bit Depth: 8
Color Depth: 'grayscale'
Format Signature: ' '
Comment: { }

Where file size is in bytes. The number of bytes in the original image is corrupted simply by multiplying width by height by bit depth and dividing the result by 8. The result is 486948.Dividing this file size gives the compression ratio:(486948/13849)=35.16.This compression ratio was achieved. While maintaining image quality consistent with the requirements of the appearance. In addition to the obvious advantages in storage space, this reduction allows the transmission of approximately 35 times the amount of uncompressed data per unit time.

The information fields displayed by imfinfo can be captured into a so called structure variable that can be for subsequent computations. Using the receding an example and assigning the name K to the structure variable.

We use the syntax

>>K=imfinfo ('bubbles25.jpg');

To store in to variable K all the information generated by command imfinfo, the information generated by imfinfo is appended to the structure variable by means of fields, separated from K by a dot. For example, the image height and width are now stored in structured fields K. Height and K. Width.

As an illustration, consider the following use of structure variable K to commute the compression ratio for bubbles25.jpg:

```
>> K=imfinfo('bubbles25.jpg');
>> image_ bytes =K.Width* K.Height* K.Bit Depth /8;
>> Compressed_ bytes = K.FilesSize;
>> Compression_ ratio=35.162
```

Note that iminfo was used in two different ways. The first was t type imfinfo bubbles25.jpg at the prompt, which resulted in the information being displayed on the screen. The second was to type K=imfinfo ('bubbles25.jpg'),which resulted in the information generated by imfinfo being stored in K. These two different ways of calling imfinfo are an example of command_ function duality, an important concept that is explained in more detail in the MATLAB online documentation.

More general imwrite syntax applicable only to tif images has the form

Imwrite(g,'filename.tif','compression','parameter',….'resloution',[colres rowers])

Where 'parameter' can have one of the following principal values: 'none' indicates no compression, 'pack bits' indicates pack bits compression (the default for non 'binary images') and 'ccitt' indicates ccitt compression. (the default for binary images).The 1*2 array [colres rowers] contains two integers that give the column resolution and row resolution in dot per_ unit (the default values). For example, if the image dimensions are in inches, colres is in the number of dots(pixels)per inch (dpi) in the vertical direction and similarly for rowers in the horizontal direction. Specifying the resolution by single scalar, res is equivalent to writing [res res].

```
>>imwrite(f,'sf.tif','compression','none','resolution',……..[300 300])
```

the values of the vector[colures rows] were determined by multiplying 200 dpi by the ratio 2.25/1.5, which gives 30 dpi. Rather than do the computation manually, we could write

```
>> res=round(200*2.25/1.5);
>>imwrite(f,'sf.tif','compression','none','resolution',res)
```

where its argument to the nearest integer.It function round rounds is important to note that the number of pixels was not changed by these commands. Only the scale of the image changed. The original 450*450 image at 200 dpi is of size 2.25*2.25 inches. The new 300_dpi image is identical, except that is 450*450 pixels are distributed over a 1.5*1.5_inch area. Processes such as this are useful for controlling the size of an image in a printed document with out sacrificing resolution.

Often it is necessary to export images to disk the way they appear on the MATLAB desktop. This is especially true with plots .The contents of a figure window can be exported to disk in two ways. The first is to use the file pull-down menu is in the figure window and then choose export. With this option the user can select a location, filename, and format. More control over export parameters is obtained by using print command:

Print-fno-dfileformat-rresno filename

Where no refers to the figure number in the figure window interest, file format refers one of the file formats in table above. 'resno' is the resolution in dpi, and filename is the name we wish to assign the file.

If we simply type print at the prompt, MATLAB prints (to the default printer) the contents of the last figure window displayed. It is possible also to specify other options with print, such as specific printing device.

3.1. Data Classes

Although we work with integers coordinates the values of pixels themselves are not restricted to be integers in MATLAB. Table above list various data classes supported by MATLAB and IPT are representing pixels values. The first eight entries in the table are refers to as numeric data classes. The ninth entry is the char class and, as shown, the last entry is referred to as logical data class.

All numeric computations in MATLAB are done in double quantities, so this is also a frequent data class encounter in image processing applications. Class unit 8 also is encountered frequently, especially when reading data from storage devices, as 8 bit images are most common representations found in practice. These two data classes, classes logical, and, to a lesser degree, class unit 16 constitute the primary data classes on which we focus. Many ipt functions however support all the data classes listed in table. Data class double requires 8 bytes to represent a number uint8 and int 8 require one byte each, uint16 and int16 requires 2bytes and unit 32.

3.2.Name And Description

Double = Double _ precision, floating_ point numbers the Approximate.
Uinit8 = unsigned 8_bit integers in the range [0,255] (1byte per Element.)
Uinit16 = unsigned 16_bit integers in the range [0,65535] (2byte Per element).
Uinit 32 = unsigned 32_bit integers in the range [0,4294967295] (4 bytes per element).
Int8 = signed 8_bit integers in the range [-128,127] (1 byte per element).
Int 16 = signed 16_byte integers in the range [32768, 32767] (2 bytes per element).
Int 32 = Signed 32_byte integers in the rang [-2147483648, 21474833647] (4 byte perElement).
Single = single _precision floating _point numbers with values in the approximate range (4 bytes per elements).
Char = characters (2 bytes per elements).
Logical values are 0 to 1 (1byte per element).

int 32 and single, requires 4 bytes each. The char data class holds characters in Unicode representation. A character string is merely a 1*n array of characters logical array contains only the values 0 to 1,with each element being stored in memory using function logical or by using relational operators.

3.3.Image Types

The toolbox supports four types of images:

- Intensity images
- Binary images
- Indexed images
- R G B images

Most monochrome image processing operations are carried out using binary or intensity images, so our initial focus is on these two image types. Indexed and RGB colour images.

3.3.1. Intensity Images

An intensity image is a data matrix whose values has been scaled to represent intentions. When the elements of an intensity image are of class unit8, or class unit 16, they have integer values in the range [0,255] and [0, 65535], respectively. If the image is of class double, the values are floating _point numbers. Values of scaled, double intensity images are in the range [0, 1] by convention.

3.3.2.Binary Images

Binary images have a very specific meaning in MATLAB.A binary image is a logical array 0s and1s.Thus, an array of 0s and 1s whose values are of data class, say unit8, is not considered as a binary image in MATLAB .A numeric array is converted to binary using function logical. Thus, if A is a numeric array consisting of 0s and 1s, we create an array B using the statement.

B=logical (A)

If A contains elements other than 0s and 1s.Use of the logical function converts all nonzero quantities to logical 1s and all entries with value 0 to logical 0s.

Using relational and logical operators also creates logical arrays.

To test if an array is logical we use the I logical function:

islogical(c)

If c is a logical array, this function returns a 1.Otherwise returns a 0. Logical array can be converted to numeric arrays using the data class conversion functions.

3.3.3.Indexed Images

An indexed image has two components:

A data matrix integer, x.

A color map matrix, map.

Matrix map is an m*3 arrays of class double containing floating_ point values in the range [0, 1].The length m of the map are equal to the number of colors it defines. Each row of map specifies the red, green and blue components of a single colour. An indexed image uses "direct mapping" of pixel intensity values colour map values. The colour of each pixel is determined by using the corresponding value the integer matrix x as a pointer into the map. If x is of class double ,then all of its components with values less than or equal to 1 point to the first row in map, all components with value 2 point to the second row and so on. If x is of class units or unit 16, then all components value 0 point to the first row in map, all components with value 1 point to the second and so on.

3.3.4. RGB Image

An RGB colour image is an M*N*3 array of colour pixels where each colour pixel is a triplet corresponding to the red, green and blue components of an RGB image, at a specific spatial location. An RGB image may be viewed as "stack" of three gray scale images that when fed in to the red, green and blue inputs of a color monitor.

Produce a color image on the screen. Convention the three images forming an RGB colour image are referred to as the red, green and blue component images. The data class of the component images determines their range of values. If an RGB image is of class double the range of values is [0, 1].

Similarly the range of values is [0,255] or [0, 65535].For RGB images of class units or unit 16 respectively. The number of bits use to represents the pixel values of the component images determines the bit depth of an RGB image. For example, if each component image is an 8bit image, the corresponding RGB image is said to be 24 bits deep.

Generally, the number of bits in all component images is the same. In this case the number of possible colours in an RGB image is (2^b) ^3, where b is a number of bits in each component image. For the 8bit case the number is 16,777,216 colors.

4.Results & Discussion

Digital Image processing is a topic of great relevance for practically any project, either for basic arrays of photodetectors or complex robotic systems using artificial vision. It is an interesting topic that offers to multimodal systems the capacity to see and understand their environment in order to interact in a natural and more efficient way. The development of new equipment for high speed image acquisition and with higher resolutions requires a significant effort to develop techniques that process the images in a more efficient way. In addition, medical applications use new image modalities and require algorithms for the interpretation of these images as well as for the registration and fusion of the different modalities, such that the image processing becomes a highly productive area for the development of multidisciplinary applications.

The aim of this research paper is to present different digital image processing algorithms using LabView and IMAQ vision toolbox. IMAQ vision toolbox presents a complete set of digital image processing and acquisition functions that improve the efficiency of the projects and reduce the programming effort of the users obtaining better results in a shorter time.

Therefore, the IMAQ vision toolbox of LabView in the opinion of the authors is an interesting tool to analyse the research work in detail and through this research paper it has been presented in order to design and implement digital image processing and develop it for different applications in the field of image acquisition and image transformations. The results of this research paper include in the first place the image acquisition and some of the most common operations that can be locally or globally applied and also the statistical information generated by the image in a histogram is utilized later in this research work. Finally, the use of tools allowing to segment or filtrate the image is highlighted making special emphasis on the algorithms of pattern recognition and matching template which greatly helped us in the successful preparation and presentation of this research work.

5.References

1. Averbuch, R. R.s Coifman, D. L. Donoho, M. Israeli, and J. Waldén, "Polar FFT, rectopolar FFT, and applications," Stanford Univ., Stanford, CA, Tech. Rep., 2000.
2. E. J. Candès, "Harmonic analysis of neural netwoks," Appl. Comput.Harmon. Anal., vol. 6, pp. 197–218, 1999.
3. "Monoscale ridgelets for the representation of images with edges,"Dept. Statist., Stanford Univ., Stanford, CA, Tech. Rep., 1999, submitted for publication.
4. "On the representation of mutilated Sobolev functions," Dept. Statist., Stanford Univ., Stanford, CA, Tech. Rep., 1999.
5. E. J. Candès and D. L. Donoho, "Curvelets," [Online] Available:http://www-tat.stanford.edu/~donoho/Reports/1999/curvelets.pdf,1999.
6. "Curvelets—A surprisingly effective nonadaptive representationfor objects with edges," in Curve and Surface Fitting: Saint-Malo 1999,A. Cohen, C. Rabut, and L. L. Schumaker, Eds. Nashville, TN: Vanderbilt Univ. Press, 1999.
7. "Ridgelets: The key to higher-dimensional intermittency?," Phil.Trans. R. Soc. Lond. A., vol. 357, pp. 2495–2509, 1999.