Design of face recognition image processor

Abstract

This project deals with the design and implementation of an image processing system for Face recognition using MATLAB. Image treatment is a complex task so, we must study all the background information that image formation and processing requires, and learn the main MATLAB functions which will have to be used.

The purpose for this study is to investigate a software application that can show how an image is processed in computer platform. The processing will be done in comparing the sketch image with the real picture to matrix model by using MATLAB program. Picture will be shown when program is running successfully. Real image or picture will be resulted from the matrix using the function in the MATLAB. We can use various functions such as filter or rotate depending on the user itself. In this study, the picture or real image used is from Internet that has referenced properly, scanner and etc. Basic mathematical calculation does not apply in this project as it only used MATLAB program. The significant of this project is to educated user and for us to learn how to process images by MATLAB to learn how the image can be changes after the function indicated by the program.

Introduction:

Our project is about to know how we can employ image processing application by using MATLAB functions. By the help of image processing Toolbox of MATLAB we were able to modify/write a program with GUI to read images, process them, blur them, and then recognize them as versions of the same images that exist in an image database; lastly we were able to display the original and blurred images.

Image processing is the field of signal processing where both the input and output signals are images. Images can be thought of as two-dimensional signals via a matrix representation, and image processing can be understood as applying standard one-dimensional signal processing techniques to two-dimensional signals. Image processing is a very important subject, and finds applications in such fields as photography, satellite imaging, medical imaging, and image compression, just to name a few.

In the past, image processing was largely done using analog devices. However, as computers have become more powerful, processing shifted toward the digital domain. Like one-dimensional digital signal processing, digital image processing overcomes traditional analog “ problems” such as noise, distortion during processing, inflexibility of system to change, and difficulty of implementation.

The image processing technique we will be implementing will be image blurring even there are many image processing techniques we have by using MATLAB to output the image as a matrix and store it in the data memory.

In today’s world, digital technology is ever growing, and the development of digitally based products is rising. Various industries such as audio, video, and cellular industry rely heavily on this digital technology. A great part of this deals with digital signal processing. This aspect in engineering has gained increasing interest, especially with much of the world now turning to wireless technology and its applications to keep businesses and industries connected. The world of digital technology is certainly one that will be present for many years to come. [Ref: 4]

Project outline:

This report consists four chapters. In first chapter, it discuss about the objective and scope of this project as long as summary of works. While Chapter 2 will discuss more on theory and literature reviews that have been done. In Chapter 3, the discussion will be on the methodology hardware and software implementation of this project. The result and discussion will be presented in Chapter 4. Last but not least, Chapter 5 discusses the conclusion of this project and future work that can be done.

Problem Statement:

In the image processing program, the info for the function are not stated clearly enough and make people understand. In the GUI (Graphical User Interface) program, the info should function as pop-up window after user press any function button.

As the project title is Image Processing using MATLAB Learning Tool, the information is not good enough and clears to understand to be recognized by people. The main problem is the effectiveness of people to recognize it. Basically we have used many techniques through which we tried to simplified the way of face recognition. We have used eigenface technique that is very standarlize way to recognize the face using MATLAB application

MATLAB also can be used in industry in the areas of bar coding, deck-top publication, copy preparation for printing and factory automation. However, due to the information and studies this state of program of image processing that I only can create. More advance and more functional program can be creating by using MATLAB. Thereby, to write the program became problem and this project not perfectly complete.

The problem which comes to set a task to recreate the convolution function for applying filters in image processing. It is very difficult to manage and get the code working. It is also not easy to write our own m-function for unsharp masking of a given image to produce a new output image.

During the project development we found following difficulties

- Apply smoothing to produce a blurred version of the original image, subtract the blurred image from the original image to produce an edge image.
- Add the edge image to the original image to produce a sharpened image.
- When carrying out the convolution image is cropped down by some pixel, this means when we go to carry out the subtraction for the unsharpening the images are not the same size and the subtraction cannot take place.

To overcome this problems we created a blank matrix in the convolution function that is the same size as the image being inputted, the new image will then go on top of this matrix so in affect the new image has a 1 pixel border around it to make it to its original size.

It is very interesting and challenging to come out from these above mentioned problems and for that we have done.

Solutions to problems in the field of digital image processing generally require extensive experimental work involving software simulation and testing with large sets of sample images. Although algorithm development typically is based on theoretical underpinnings, the actual implementation of

these algorithms almost always requires parameter estimation and, frequently, algorithm revision and comparison of solutions.

Because it works in the MATLAB computing environment, the Image Processing Toolbox offers some significant advantages

Key components of our approach

We have used Eigen Vector method [Ref 12] that is a set of eigenfaces can be generated by performing a mathematical process called principal component analysis (PCA) on a large set of images depicting different human faces. Informally, eigenfaces can be considered a set of " standardized face ingredients", derived from statistical analysis of many pictures of faces. Any human face can be considered to be a combination of these standard faces. For example, one's face might be composed of the average face plus 10% from eigenface 1, 55% from eigenface 2, and even -3% from eigenface 3. Remarkably, it does not take many eigenfaces combined together to achieve a fair approximation of most faces. Also, because a person's face is not recorded by a digital photograph, but instead as just a list of values (one value for each eigenface in the database used), much less space is taken for each person's face.

Apart from these our project methodology includes the following:

1. Use MATLAB to simulate the processing technique.
2. Carefully locating the memory blocks where we will store our original and output image.
3. Comparing our results in MATLAB.

Basically the eigenvectors of a square matrix are the non-zero vectors that, after being multiplied by the matrix, remain proportional to the original vector (i. e., change only in magnitude, not in direction). For each eigenvector, the corresponding eigenvalue is the factor by which the eigenvector changes when multiplied by the matrix.

The eigenvectors are sometimes also called proper vectors, or characteristic vectors. Similarly, the eigenvalues are also known as proper values, or characteristic values.

The mathematical expression of this idea is as follows: if A is a square matrix, a non-zero vector v is an eigenvector of A if there is a scalar λ (lambda) such that

The scalar λ is said to be the eigenvalue of A corresponding to v. An eigenspace of A is the set of all eigenvectors with the same eigenvalue together with the zero vectors. However, the zero vector is not an eigenvector. any problems present themselves in terms of an eigenvalue problem:

$A \cdot v = \lambda \cdot v$

In this equationAis an n-by-n matrix, vis a non-zero n-by-1 vector and λ is a scalar (which may be either real or complex). Any value of λ for which this equation has a solution is known as an eigenvalue of the matrixA. It is sometimes also called the characteristic value. The vector, v, which corresponds to this value is called an eigenvector. The eigenvalue problem can be rewritten as

A·v-λ·v= 0

A·v-λ·I·v= 0

(A-λ·I·v)= 0

If v is non-zero, this equation will only have a solution if

| A-λ·I|= 0

This equation is called the characteristic equation ofA, and is an nthorder polynomial in λ with n roots. These roots are called the eigenvalues ofA. We will only deal with the case of n distinct roots, though they may be repeated. For each eigenvalue there will be an eigenvector for which the eigenvalue equation is true. This is most easily demonstrated by example

Example: Find Eigenvalues and Eigenvectors of a 2×2 Matrix

If

then the characteristic equation is

and the two eigenvalues are

λ1=-1, λ2=-2

All that’s left is to find the two eigenvectors. Let’s find the eigenvector, v1, associated with the eigenvector, λ1=-1, first.

so clearly from the top row of the equations we get

Note that if we took the second row we would get

In either case we find that the first eigenvector is any 2 element column vector in which the two elements have equal magnitude and opposite sign.

Where k1is an arbitrary constant. We didn’t have to use +1 and -1, we could have used any two quantities of equal magnitude and opposite sign.

Going through the same procedure for the second eigenvalue:

Again, the choice of +1 and -2 for the eigenvector was arbitrary; only their ratio is important.

Scope of Project

The scope of our project includes the following:

Study and understand the image processing in varies method, mainly in MATLAB.

Create a GUI (Graphical User Interface) MATLAB program with several functions.

This requires identifying the steps which must be done to obtain some results. Further this project, the main areas considered are:

- Study about MATLAB, and its main functions to obtain and process images.
- Write or modify a program which can be used to acquire and treat images.
- Some information about the image file and its characteristics to understand the information it contains.

Objective of the Project

The objective of this project is actually to educate us and new comers to basic and fundamental technique in image processing through integrated image processing software. All fundamental algorithms of image processing will be exposed through this package [Ref] the program is in appendix -B. This package will also provided easy-to-learn mechanisms turn user-friendly and graphic-orientation environment.

These operations include preprocessing, spatial filtering, image enhancement, feature detection, image compression and image restoration involves process which restores a degraded image to something close to the ideal. Generally, in computer vision, especially in MATLAB program (image understanding or scene analysis) involves technique from image processing, pattern recognition and artificial intelligent. Particularly, MATLAB program offers many features and are more multifaceted then any calculator. MATLAB toolbox is a tool for making mathematical calculations.

Literature review (Related Work to our Project)

Image processing is any form of signal processing for which the input is an image, such as photographs; the output of image processing can be either an image or a set of characteristics or parameters related to the image. Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal-processing techniques to it. Image processing – converting the image to another form by using direction in MATLAB/Toolboxes/Image Processing tables for example is image input/output, color operation, image enhancement/ analysis and another method.

Image processing and computer vision practitioners tend concentrate on a particular area of specialization. People refer to their research interests as “ texture”, “ surface mapping”, “ video tracking”, and the like. Nevertheless, there is a strong need to appreciate the spectrum and hierarchy of processing levels.

Image processing is the manipulation of the image by using a computer, with the objective to enhance or evaluate some aspect of an image which is not readily apparent in its original form. This is done through the development and implementation of processing means necessary to operate on the image. Processing image using a digital computer provides the greatest flexibility and power for general image processing application, since the programming of a computer can be changed easily which allows operation to be modified quickly.

Interest in image processing technique dates back to early 1920’s when digitized pictures of world news events were first transmitted by submarine cable between Newyork and London. However, application of digital image processing concepts did not become widespread until the middle 1960’s, when third-generation digital computers began to offer the speed and storage capabilities required for practical implementation of image processing algorithms. Since then, this area has experienced vigorous growth and has been subjected of study and research in such fields as engineering, computer science, statistics, information science, physics, chemistry and medicine.

The result of these efforts have established the value of image processing technique in of problem with application in diverse fields, including automated factory controlled, astronomy, meteorology, agriculture, medicine, art and military application. With the increasing availability of reasonably inexpensive hardware and some very importance application on the horizon, image technology is expected to continue its growth and to play an important role in the future.

From the MATLAB software we have the Toolbox for image processing and Professional MATLAB. MATLAB is the interactive environment, scientists and engineers are able to analyze and develop algorithms with exceptional improvements n productivity and creativity. As a result of new algorithms with application-specific uses. The MathWorks offers a series of application toolboxes that contain set of MATLAB ofr the Linear algebra, high-speed computational kernel, extensive mathematical functionality, data analysis, 2-D and 3-D graphic rapid algorithm development, matrix based programming environment. In MATLAB Toolboxes professional version but priced at a lower rate for academic use. [Ref: 4]

About Image Processing Tools of Matlab

This set of Matlab tools consists of some functions that I have found useful for basic image processing and image analysis.

When working with binary objects (4-connected foreground regions), we have often found it useful to measure features from the boundary stored as a list of coordinates. In other words, sometimes it is better to work with a polygon defining the foreground-background boundary than to work with a

black and white image of the object. The boundary of an object in a binary (black and white) image can be stored as a list of pixel corner coordinates. The functiongetboundarymex [Ref 7] forms a list of these corner coordinates from a binary image containing an object.

The toolbox containsselectobjectmexfor selecting regions by size. The commandimOut= selectobjectmex(imIn, n)will return an image, imOut, containing only thenth largest object (in terms of number of pixels) of the original imageimIn. This function is particularly useful if one wants to quickly threshold an image and then select the largest object without having to worry about smaller objects that are not of interest, e. g. imOut= selectobjectmex(im> 0. 5, 1).

[Ref: 9]

Also included is code for watershed segmentation by flooding from selected sources, fast calculation of object centroids etc.

The usage of each the function is described by typinghelpfunctionat the MATLAB command prompt, wherefunctionis the name of the relevant function.

The M-file scriptkftools shows an example of the usage of all of the functions in this toolbox .

THEORY:

There are various ways of implementing the image blurring technique:

1. Linear blur – horizontal or vertical averaging of a fixed number of pixels.
2. Block blur – averaging a small block of pixels by propagating a fixed sized window through the entire image.
3. Gaussian blur – convolution of the image with a two-dimensional Gaussian function.

Linear blur:

This is the simplest image blurring technique. It is done by taking the N-point average of a linear block of pixels (either horizontally or vertically). In our implementation, N will be 8, and we will be using the horizontal blur. An 1×N-pixel window is placed at the top left of the image, and the average of the window is stored in the N/2th pixel of the window (in a new image to prevent overwriting). The window is then shifted across the row and the process is repeated. Once the window reaches the end of the row, it is moved to the next row and the process repeats itself. [Ref: 11]

The advantage of this method is that it is the simplest of the three. However, it also gives the poorest blurring quality. This is because by taking the horizontal average of each row, there will be averaging “ lines” in the output image. Also, parts of the picture where the detail does not span enough horizontal pixels will be lost after blurring. Finally, by the way this algorithm is designed, there will be an outer frame of the output image identical to the input image (i. e. the outer part of the image remains not blurred). [Ref: 11]

Block blur:

This method is analogous to the linear blur, except that our window is now an N×N-pixel window. The procedure is the same as the linear blur, with the averaged pixel stored in the (N/2, N/2) position of the window. See

block_blur. m for the MATLAB implementation of this algorithm.

This method improves upon the quality of the linear blur in that averaging “ lines” are no longer visible in the output image. It also helps to retain details that span small horizontal distances in the original image better. However, it still does not overcome the problem of an outer frame in the output image that remains not blurred. [Ref: 11]

Gaussian blur:

This is the best implementation of the image blurring technique, and is used in such commercial software as Adobe Photoshop. Unfortunately, it is also the most complex. It works by performing a two-dimensional convolution on the input image with a normalized two-dimensional M×M-pixel Gaussian function.

Intuitively, each pixel of the output image is actually a Gaussian function centred at each point of the input image. Hence, the convolution will increase the size of the output image to N+M-1, so that after convolution we must crop the image to reduce it to its proper size.

This method is the best of the three. It has no averaging “ lines” present, and it also blurs the entire image.

Image Processing Toolbox (give reference to the Toolbox)

Image Processing Toolbox provide us a comprehensive set of reference standard algorithms and graphical tools for image processing, analysis, visualization, and algorithm development. We can perform image enhancement, image deblurring, feature detection, noise reduction, image segmentation, spatial transformations, and image registration.

Image Processing Toolbox supports a diverse set of image types, including high dynamic range, gigapixel resolution, ICC-compliant color, and tomographic images. Graphical tools let we explore an image, examine a region of pixels, adjust the contrast, create contours or histograms, and manipulate regions of interest (ROIs). With the toolbox algorithms we can restore degraded images, detect and measure features, analyze shapes and textures, and adjust the color balance of images.

Key Features

- Image enhancement, filtering, and deblurring
- Image analysis, including segmentation, morphology, feature extraction, and measurement
- Spatial transformations and image registration
- Image transforms, including FFT, DCT, Radon, and fan-beam projection
- Workflows for processing, displaying, and navigating arbitrarily large images
- Modular interactive tools, including ROI selections, histograms, and distance measurements
- ICC color management
- Multidimensional image processing

- Image-sequence and video display
- DICOM import and export

We have collected many image processing function which can make our project easy to execute , some of these function we used are as follows.

Image Display and Exploration

Immovie:	Make movie from multiframe image
Implay:	Play movies, videos, or image sequences
Imshow:	Display image
Imtool:	Image Tool
Montage:	Display multiple image frames as rectangular montage
Subimage:	Display multiple

	images in single figure
Warp:	Display image as texture-mapped surface

Image File I/O

analyze75info:	Read metadata from header file of Analyze 7. 5 data set
analyze75read:	Read image data from image file of Analyze 7. 5 data set
Dicomanon:	Anonymize DICOM

	file
Dicomdict:	Get or set active DICOM data dictionary
Dicominfo:	Read metadata from DICOM message
Dicomlookup:	Find attribute in DICOM data dictionary
dicomread:	Read DICOM image
Dicomuid:	Generate DICOM unique

	identifier
Dicomwrite:	Write images as DICOM files
Hdrread:	Read high dynamic range (HDR) image
Hdrwrite:	Write Radiance high dynamic range (HDR) image file
Interfileinfo:	Read metadata from Interfile file
Interfilerea	Read

d:	images in Interfile format
Isrset:	Check if file is R-Set
Makehdr:	Create high dynamic range image
Nitfinfo:	Read metadata from National Imagery Transmission Format (NITF) file
Nitfread:	Read image from NITF file

Openrset:	Open R-Set file
Rsetwrite:	Create reduced resolution data set from image file

Image Types and Type Conversions

Demosaic:	Convert Bayer pattern encoded image to truecolor image
gray2ind:	Convert grayscale or binary image to indexed image
Grayslice:	Convert grayscale image to

	indexed image using multilevel thresholding
Graythresh:	Global image threshold using Otsu’s method
im2bw:	Convert image to binary image, based on threshold
im2double:	Convert image to double precision
im2int16:	Convert image to 16-bit signed integers
im2java	Convert

	image to
2d:	Java buffered
	image
	Convert
im2singl	image to
e:	single
	precision
	Convert
im2uint1	image to 16-
6:	bit unsigned
	integers
	Convert
im2uint8	image to 8-
:	bit unsigned
	integers
	Convert
	indexed
ind2gray	image to
:	grayscale
	image
	Convert
ind2rgb:	indexed
	image to

	RGB image
label2rgb:	Convert label matrix into RGB image
mat2gray:	Convert matrix to grayscale image
rgb2gray:	Convert RGB image or colormap to grayscale

We both studied the function properly and found few of them are very important for us to understand deeply. In order to segregate the most important function we select some of from them.

System description:

This project will use the MATLAB software package to develop algorithms which can automatically analyze these images for potential comets. MATLAB is a high-level programming environment very popular with scientists and engineers because of its powerful toolboxes and easy to use scripting language. Basic algorithms from the image processing toolbox will be utilized to find comets using the following general steps:

1. Load original images into MATLAB
2. Process images to isolate all bright spots and eliminate glare due to solar ejections
3. Compare spots in subsequent images to find potential comet trajectories
4. Analyze trajectories to ensure they meet known characteristics
5. Highlight possible comets in original images and create output image

Basically, MATLAB software has many functions/commands to apply in image processing. How to manipulate the program depending to us but must be practically know what item is MATLAB program will be used. Creativity in MATLAB can make the interesting result. Even, the complex data can be solved in MATLAB. Especially when the data involved is very complex. Here, we can create some image from converting data by using the some program in MATLAB, which just applied all procedure in the MATLAB program. MATLAB toolbox is a tool for making mathematical calculations. Image processing toolbox is user friendly programming language with feature more advanced. In the program also used the GUI (Graphical User Interface, move this definition to the first place where we used GUI) to create develop the program.

Techniques and algorithm:

Image and MATLAB involves the conversion of scene into a digital representation that can be processed by a digital computer. This can be performed by a sensor system specially designed to view a image and provide a digital representation of the image.

When the images are installed in MATLAB, my picture for example, the color of that image is first analyzed. In the process include several functions of image processing technique. Processed Image is the image display after the process.

GUI (Graphical User Interface)

A graphical user interface (GUI) is a graphical display in one or more windows containing controls, called components that enable a user to perform interactive tasks. The user of the GUI does not have to create a script or type commands at the command line to accomplish the tasks. Unlike coding programs to accomplish tasks, the user of a GUI need not understand the details of how the tasks are performed.

GUI components can include menus, toolbars, push buttons, radio buttons, list boxes, and sliders just to name a few. GUIs created using MATLAB tools can also perform any type of computation, read and write data files, communicate with other GUIs, and display data as tables or as plots

Most GUIs wait for their user to manipulate a control, and then respond to each action in turn. Each control, and the GUI itself, has one or more user-written routines (executable MATLAB code) known as callbacks, named for the fact that they “ call back” to MATLAB to ask it to do things. The execution of each callback is triggered by a particular user action such as pressing a screen button, clicking a mouse button, selecting a menu item, typing a string or a numeric value, or passing the cursor over a component. The GUI then responds to these events. We, as the creator of the GUI, provide callbacks which define what the components do to handle events.

This kind of programming is often referred to as event-driven programming. In the example, a button click is one such event. In e vent-driven programming, callback execution is asynchronous, that is, it is triggered by events external to the software. In the case of MATLAB GUIs, most events are user interactions with the GUI, but the GUI can respond to other kinds of events as well, for example, the creation of a file or connecting a device to the computer.

We can code callbacks in two distinct ways:

- As MATLAB functions, written in M and stored in M-files
- As strings containing MATLAB expressions or commands (such as ‘ c = sqrt(a*a + b*b);’or ‘ print’)

Using functions stored in M-files as callbacks is preferable to using strings, as functions have access to arguments and are more powerful and flexible. MATLAB scripts (sequences of statements stored in M-files that do not define functions) cannot b e used as callbacks.

Although we can provide a callback with certain data and make it do anything we want, we cannot control when callbacks will execute. That is, when wer GUI is being used, we have no control over the sequence of events that trigger particular callbacks or what other callbacks might still be running at those times. This distinguishes event-driven programming from