CHAPTER-1

INTRODUCTION

1.1 MATLAB DESIGN

MATLAB (an abbreviation of "MATrix LABoratory") is a proprietary multi paradigm programming language and numeric computing environment developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages. Although MATLAB is intended primarily for numeric computing, an optional toolbox uses the MuPAD symbolic engine allowing access to symbolic computing abilities.

As of 2020, MATLABs has more than four million users worldwide. They come from various backgrounds of engineering, science, and economics. As of 2017, more than 5000 global colleges and universities use MATLAB to support instruction and research.

1.2 HISTORY OF MATLAB

MATLAB was invented by mathematician and computer programmer Cleve Moler. The idea for MATLAB was based on his 1960s PhD thesis. Moler became a math professor at the University of New Mexico and started developing MATLAB for his students as a hobby.

He developed MATLAB's initial linear algebra programming in 1967 with his one-time thesis advisor, George Forsythe. This was followed by Fortran code for linear equations in 1971.

Before version 1.0, MATLAB "was not a programming language; it was a simple interactive matrix calculator. There were no programs, no toolboxes, no graphics. And no ODEs or FFTs."

The first early version of MATLAB was completed in the late 1970s. The software was disclosed to the public for the first time in February 1979 at the Naval Postgraduate School in California. Early versions of MATLAB were simple matrix calculators with 71 pre-built functions. At the time, MATLAB was distributed for free to universities. Moler would leave copies at universities he visited and the software developed a strong following in the math departments of university campuses.

In the 1980s, Cleve Moler met John N. Little. They decided to reprogram MATLAB in C and market it for the IBM desktops that were replacing mainframe computers at the time.

MATLAB was first released as a commercial product in 1984 at the Automatic Control Conference in Las Vegas. MathWorks, Inc. was founded to develop the software and the MATLAB programming language was released. The first MATLAB sale was the following year, when Nick Trefethen from the Massachusetts Institute of Technology bought ten copies.

By the end of the 1980s, several hundred copies of MATLAB had been sold to universities for student use.] The software was popularized largely thanks to toolboxes created by experts in various fields for performing specialized mathematical tasks. Many of the toolboxes were developed as a result of Stanford students that used MATLAB in academia, then brought the software with them to the private sector.

Over time, MATLAB was re-written for early operating systems created by Digital Equipment Corporation, VAX, Sun Microsystems, and for Unix PCs. Version 3 was released in 1987. The first MATLAB compiler was developed by Stephen C. Johnson in the 1990s.

In 2000, MathWorks added a Fortran-based library for linear algebra in MATLAB 6, replacing the software's original LINPACK and EISPACK subroutines that were in C. MATLAB's Parallel Computing Toolbox was released at the 2004 Supercomputing Conference and support for graphics processing units (GPUs) was added to it in 2010.

Some especially large changes to the software were made with version 8 in 2012. The user interface was reworked and Simulink's functionality was expanded. By 2016, MATLAB had introduced several technical and user interface improvements, including the MATLAB Live Editor notebook, and other features.

1.3 SOFTWARE TOOLS

1.3.1 Software required

MATLAB Software

1.3.2 Introduction to MATLAB

The name MATLAB stands for matrix laboratory. It was invented in the late 1970s by Cleve Molar, then chairman of the computer science department at the University of New Mexico.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development and analysis. MATLAB was first adopted by control design engineers, Little's specialty, but quickly spread to many other domains.

It is now also used in education, in particular the teaching of linear algebra and numerical analysis, and is popular amongst scientists involved with image processing.

MATLAB is a high-performance language for technical computing. It integrates computation, programming and visualization in a user-friendly environment where problems and solutions are expressed in an easy-to-understand mathematical notation. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows the user to solve many technical computing problems, especially those with matrix and vector operations, in less time than it would take to write a program in a scalar non-interactive language such as C or FORTRAN.

MATLAB is a matrix-based programming tool. Although matrices often need not to be dimensioned explicitly, the user has always to look carefully for matrix dimensions. If it is not defined otherwise, the standard matrix exhibits two dimensions nxm. Column vectors and row vectors are represented consistently by nx1 and 1xn matrices respectively.

1.3.3 Expressions

Like most other programming languages, mat lab provides mathematical expressions, but unlike most programming languages these expressions involve entire matrices. The building blocks of expressions are Variables , Numbers ,Operators , Functions

1.3.4 Handling matrices

MATLAB was mainly designed to deal with matrices in MATLAB a matrix is a rectangular array of numbers so scalars can be interpreted to be 1x1 matrices and vectors are matrices with only one row or column. MATLAB has other ways to store both numeric and non numeric data, but in the beginning of learning MATLAB, it is usually best to think of everything as a matrix. The operations in MATLAB are designed to be as natural as possible. Where other programming languages work only with single numbers, MATLAB allows with entire matrices quickly and easily.

1.3.5 Entering matrices and addressing the elements

The elements of matrix must be entered one-by-one in a list where the elements of row must be separated with commas or blank spaces and rows are divided by semicolons.

The whole list must be surrounded with square brackets, e.g.

```
>> A= [1 2 3;8 6 4;3 6 9]
```

After pressing “Enter” Matlab displays the numbers entered in the command line

```
A =     1 2 3
        8 6 4
        3 6 9
```

Addressing an element of a matrix is also very easy. The n-th element of the m-th column in a matrix A from above is A(n,m). So typing

```
>>A(1,3)+A(2,1)+A(3,2)
```

Will compute the answer ans=17

The k-th to l-th elements of the m-th to n-th columns can be addressed by A(k:l,m:n), e.g.

```
>>A(2:3,1:2)
```

```
Ans =  8  6
       3  6
```

Further examples:

```
>>A(1,1:2)
```

Addresses the first two elements of the first row.

```
Ans=12
```

>>A(:,2) =>Addresses all elements of second column.

```
Ans =   8
        6
        4
```

1.3.6 Generating matrices

There are different ways to generate matrices. Assigning elements explicitly was presented in the above topic. To create a row vector with 101 equidistant values starting at 0 and ending by -, this method would be very tedious. So two other possibilities are shown below

```
>>X=linspace(0,pi,101)
```

Or

```
>>X=(0:0.01:1)*pi
```

In the first case, MATLAB function linspace is used to create X. The functions arguments are described by:

Linspace (first value, last value, number of values)

With the default number of values =100.

In the second case, the colon notation(0:0.01:1) creates an array that starts at 0, increments by 0.01 and ends at 1. Afterwards each element in this array is multiplied by pi to create the desired values in X.

Both of these array creation forms are common in MATLAB. While the colon notation form allows to specify the increment between data elements directly, but not the number of elements, the MATLAB function linspace allows to specify the number of data elements directly, but not the increment value between these data elements

The colon notation is very often used in MATLAB; therefore a closer look should be taken on it. (first value: increment: last value) creates an array starting at first value, ending at last value with an increment which can be negative as well, e.g.

```
>>V= (10:-2:0)
V=10 8 6 4 2 0
```

If the increment is 1, then its usage is optional:

```
>>W= (5:10)
W=5 6 7 8 9 10.
```

MATLAB also provides four functions that generates basic matrices: zeros, ones, rand and randn. Some examples

```
>>B= zeros(3,4)
B= 0 0 0 0
   0 0 0 0
   0 0 0 0
>>C= ones(2,5)*6
C=6 6 6 6 6
  6 6 6 6 6
```

>>D= rand(1,5) generates uniformly distributed random elements.

```
D=0.5028 0.7095 0.4289 0.3046 0.1897
```

>>E= randn(3,3)generates normally –also called Gaussian – distributed random elements.

```
E= -0.4326  0.2877  1.1892
   -1.6656 -1.1465 -0.0376
    0.12531909   0.3273
```

1.3.7 Concatenation

Concatenation is the process of joining small matrices to make bigger ones. In fact, the first matrix A was created by concatenating its individual elements. The pair of square brackets, [ ], is the Concatenation operator. For an example, start with the 3x3 matrix A

form

```
>>F= [A A+10;A*2 A*4].
```

The result is a 6x6 matrix, obtained by joining the four submatrices.

```
F= 1   2   3  11 12 13
   8   6   4  18 16 14
   3   6   9  13 16 19
   2   4   6   4   8  12
   16  2  8  32 24  16

   6  12 19 12 24  38
```

1.3.8 File types

MATLAB has three types of files for storing information.M-files are standard ASCII text files, with .m extension to the file name. There are two types of these files: script file and function files. Most of the programs we write in mat lab are saved as M-files. All built in functions in matlab are M-files, most of which reside on our computer in precompiled format. Some built-in functions are provided with source code in readable M-files so that they can be copied and modified. Mat-files are binary data files, with mat extension to the file name. Mat-files are created by mat lab when we save data with the save command. The data is written in a special format that only mat lab can read. Mat-files can be loaded in to mat lab with the load command.

1.3.9 M-files

In mat lab we write our programs in M-file. M-files are ordinary ASCII text files written in mat lab's language. They are called M-files because they must have ' m' at the end of their name. M-files can be created using any editor or word processing applications. There are two types of M-files –script files and function files.

A script file is an M-file with a set of valid mat lab commands in it. A script file is executed by typing the name of the file and the command line. It is equivalent to typing all the commands stored in the script file, one by one, at the mat lab prompt. Naturally, script files work on global variables, i.e variables currently present in the workspace. Results obtained from executing script files are left in the workspace. A script file may contain any number of commands including those that call built-in functions or function written by us. Script files are useful when we have to repeat the set of commands several times.

A function file is also an M-file like a script file, except that the variables in the function file are all local. Function files are like programs or subroutines in Fortran, procedures in Pascal, and functions in C. A function file begins with a function definition line, which has a well defined list of inputs and outputs. Without this line, the file becomes a script file.

The commands…between for and the end statements are executed one time for every column in array. For

Example

```
>> for k = 1:10
       z(k)=2*I;
End
>>z
z = [2 4 6 8 10 12 14 16 18 20]
```

it is good idea to indent the loops for readability, especially when they are nested.

```
>>for 1=1.5
      for m=1:8
      H(1,m) =1/1+m;
      end
```

1.4 Working with MATLAB

For simple problems, entering requests at the MATLAB prompt in the command window is fast and efficient. However, as the number of commands increases, or whenever a change of value of one or more variables with a revaluation is desired, typing at the MATLAB prompt becomes tedious. MATLAB allows to place MATLAB commands in a simple text file, and by telling MATLAB to open this file, the stored commands are evaluated one-by-one as if they were just typed in. those files are called m-files. There are two kinds of m-files:

Internal variables are local to the function. The only difference in the syntax of a script-file and a function-file is the first line. The first line in function-file starts with the keyword function followed by the list of output arguments, an equals sign, the name of the function and ending with the list of input variables enclosed in parentheses and separated by commas. If the function has multiple output arguments, the output argument list must be enclosed in a square brackets,

e.g.: Function[x,y,z]=cosytrans(theta, phi,rho)

In a script-file there is no predefined syntax for the first line.

To create or edit an M- file in the environment of Linux. The command nedit can be used to start a text editor. The command nedit must be typed at a Linux command window. The use of this editor is very simple and the most important commands can be found in top-down-menus.

Another important aspect the work with MATLAB should be mentioned here. The commands presented in this paper were introduced without concluding semicolon. Therefore a response to the commands occurs at the command prompt. So entering a new variable causes a repetition of the variable name and its values. Sometimes it is most better avoid this repetition especially in large M files. Since a load of information would appear on the screen and the really interesting data might get lost within this load. To suppress this response a concluding semicolon must be entered after the command.

1.5 Graphical user interface

The MATLAB Graphical User Interface development environment provides a set of tools for creating graphical user interfaces (GUIs).These tools greatly simplify the process of designing and building GUIs. Use the GUIDE tools to lay out the GUI Using the GUIDE Layout Editor, you can lay out a GUI easily by clicking and dragging GUI components -- such as panels, buttons, text fields, sliders, menus, and so on -- into the layout area. Program the GUI GUIDE automatically generates an M-file that controls how the GUI operates. The M-file initializes the GUI and contains a framework for all the GUI callbacks -- the commands that are executed when a user clicks a GUI component. Using the M-file editor, we can add code to the callbacks to perform the functions we want them to.

To start GUIDE, enter guide at the MATLAB prompt. This displays the GUIDE Quick Start dialog, as shown in the following figure.From the Quick Start dialog, we can create a new GUI from one of the GUIDE templates prebuilt GUIs that we can modify for our own purposes. Open an existing GUI.Select Blank GUI and press ok, this opens the layout editor.

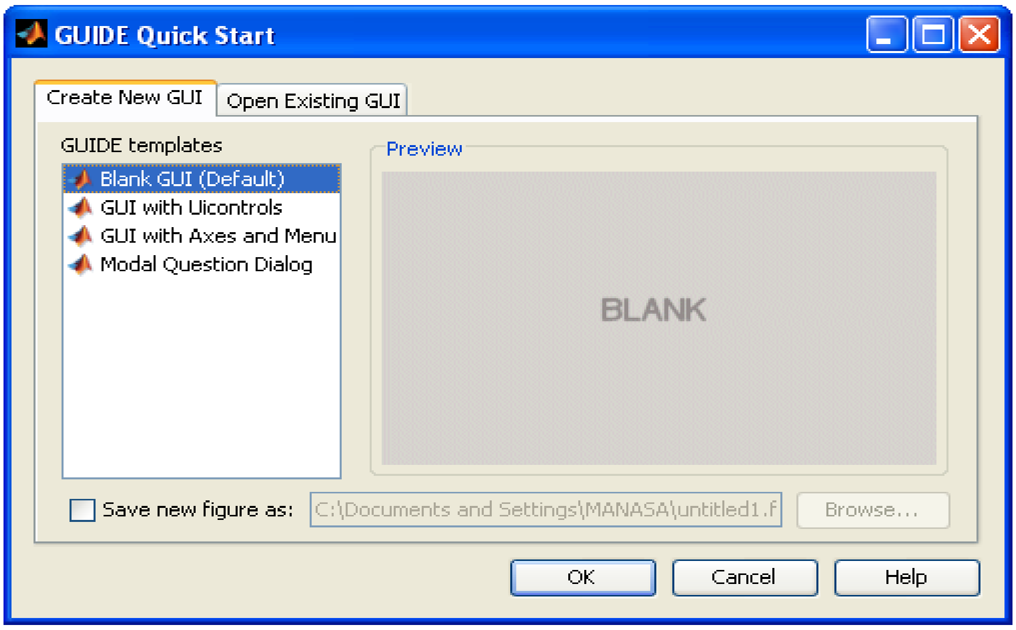


Fig.1.5 GUI quick start

When you open a GUI in GUIDE, it is displayed in the Layout Editor, which is the control panel for all of the GUIDE tools. The following figure shows the Layout Editor with a blank GUI template. We can lay out our GUI by dragging components, such as push buttons, pop-up menus, or axes, from the component palette, at the left side of the Layout Editor, into the layout area. For example, if we drag a panel into the layout area, it appears as in the following figure. We can also use the Layout Editor to set basic properties of the GUI components. Other components include push button, slider, radio button, check box, edit text, static text, popup menu, list box, Toggle button, axes, Button group, Active X group.

1.5.1 Axes

Axes are used to obtain an image from the video into a particular region which is created with certain dimensions.

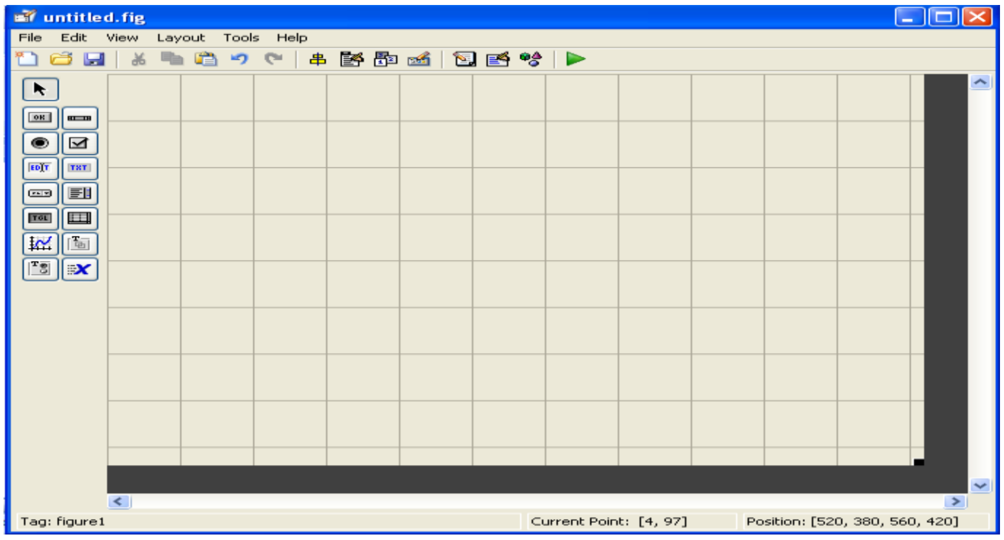

untitled.fig
File Edit View Layout Tools Help
Tag: figure1
Current Point: [4, 97]
Position: [520, 380, 560, 420]

Fig.1.5.1 Axes

1.5.2 Creation of push button

This acts as a button by which the alignment of image is selected.

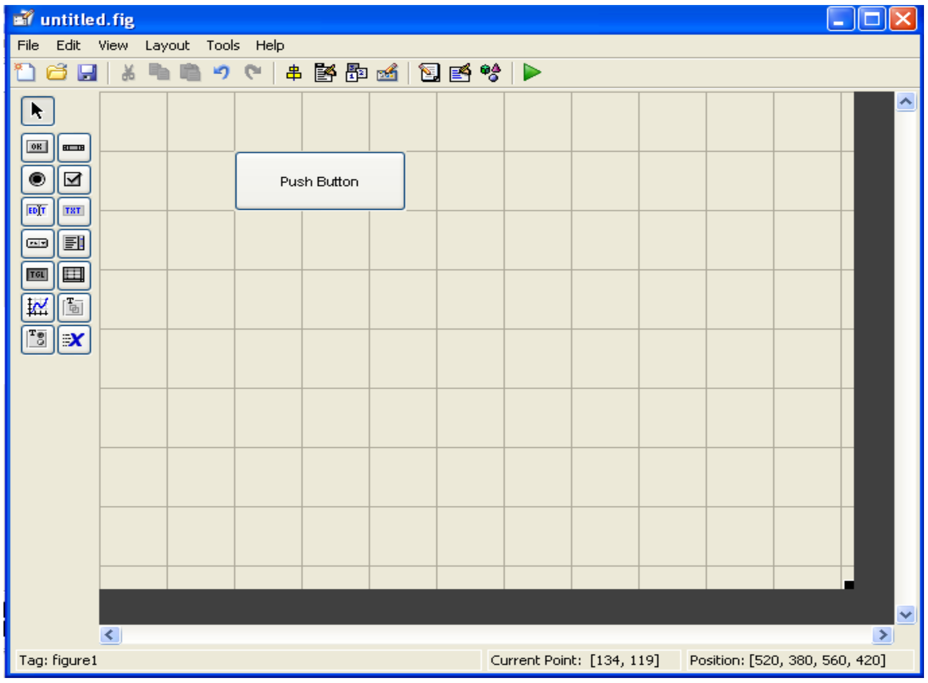
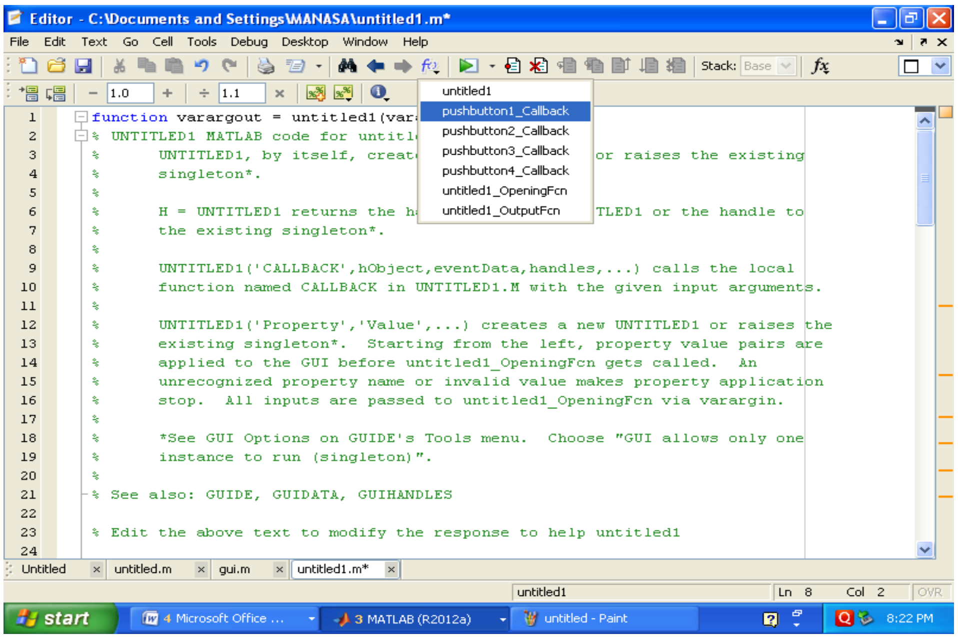

untitled.fig
File Edit View Layout Tools Help
Push Button
Tag: figure1
Current Point: [134, 119]
Position: [520, 380, 560, 420]

Fig.1.5.2 Creation of push button

1.5.3 Go for callback

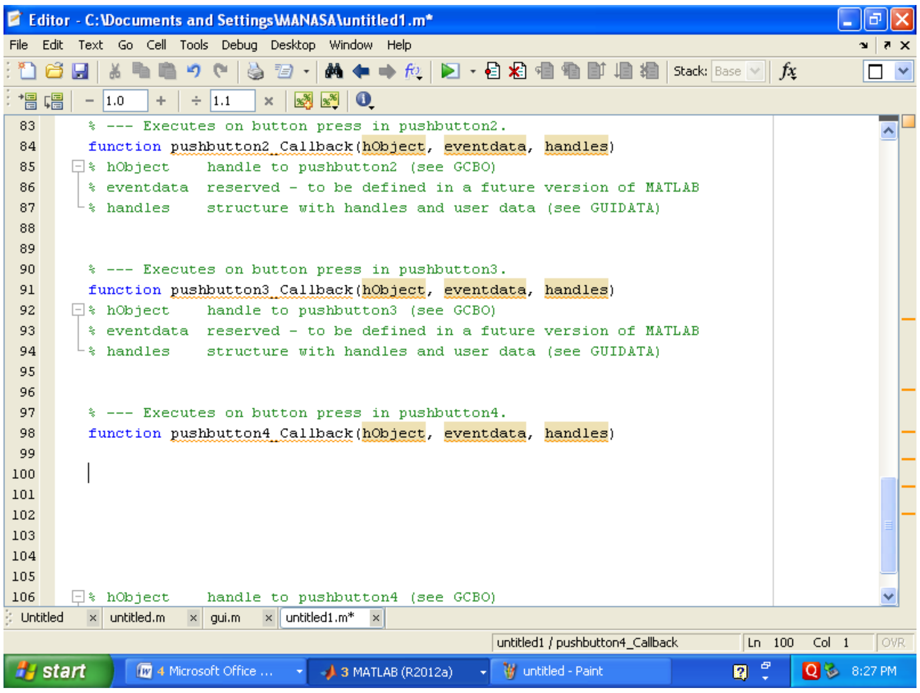
By pressing f we can know where the callback is and we can write code below the callback.



```
function varargout = untitled1(var
% UNTITLED1 MATLAB code for untitl
%      UNTITLED1, by itself, creat                     or raises the existing
%      singleton*.
%
%      H = UNTITLED1 returns the h                     TLED1 or the handle to
%      the existing singleton*.
%
%      UNTITLED1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in UNTITLED1.M with the given input arguments.
%
%      UNTITLED1('Property','Value',...) creates a new UNTITLED1 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before untitled1_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to untitled1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help untitled1
```

Fig.1.5.3 Go for callback

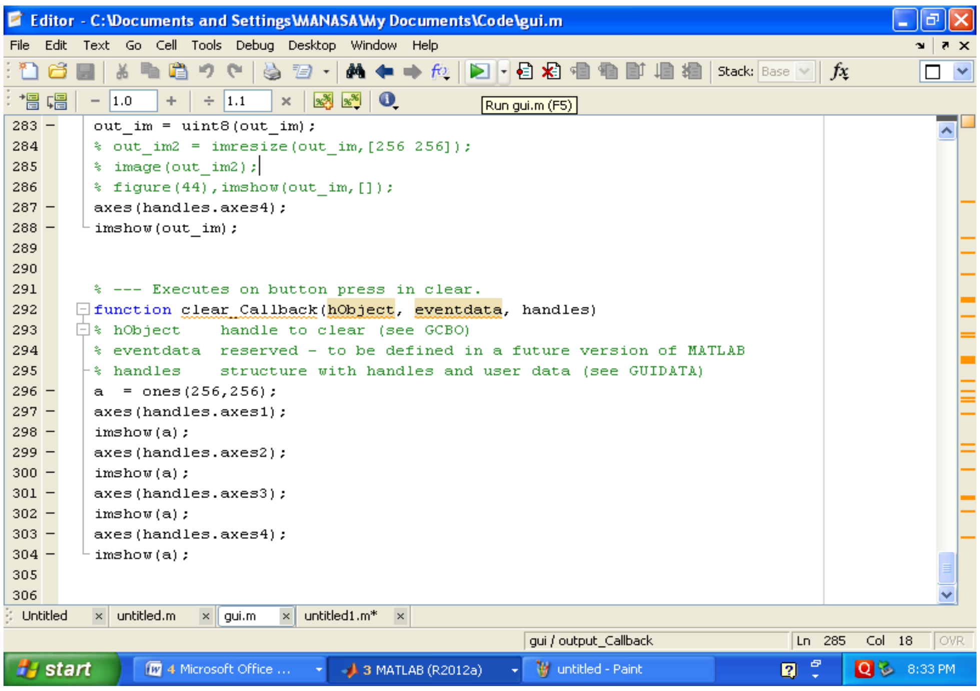
1.5.4 Write the code below callback


Editor - C:\Documents and Settings\MANASA\untitled1.m*
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
Untitled
untitled.m
gui.m
untitled1.m*
untitled1 / pushbutton4_Callback
Ln 100 Col 1 OVR
start
4 Microsoft Office ...
3 MATLAB (R2012a)
untitled - Paint
8:27 PM

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)








% hObject    handle to pushbutton4 (see GCBO)
```

Fig.1.5.4 Write the code below callback

1.5.5 RUN THE PROGRAM



```
out_im = uint8(out_im);
% out_im2 = imresize(out_im,[256 256]);
% image(out_im2);
% figure(44),imshow(out_im,[]);
axes(handles.axes4);
imshow(out_im);


% --- Executes on button press in clear.
function clear_Callback(hObject, eventdata, handles)
% hObject    handle to clear (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a  = ones(256,256);
axes(handles.axes1);
imshow(a);
axes(handles.axes2);
imshow(a);
axes(handles.axes3);
imshow(a);
axes(handles.axes4);
imshow(a);
```

Fig.1.5.5 Run the program

1.6 PROGRAMMING GUI

When we first save or run the GUI, GUIDE generates a function M-file that contains the most commonly used callbacks for each component. It also contains some initialization code, an opening function callback, and an output function callback. Each call back is a sub function that initially consists of a framework that contains just a function definition. We must add code to the callbacks to make them work. We can save a GUI by selecting Save or Save as from the File menu, or by clicking the Save icon on the toolbar. We can run the GUI by selecting Run from the Tools menu or by clicking the Run icon on the toolbar. After GUIDE generates the M-file, it opens the Save GUI as dialog. Type a name in the File name field. GUIDE assigns the same name to FIG-file and the M-file. When we click Save, GUIDE saves the M-file and open it in the M-file Editor.

When the GUI is completed and running and a user clicks a user interface control, such as a push button, MATLAB executes the callback specified by the component's Callback property Push Button Callbacks Each of the push buttons creates a different type of plot using the data specified by the current selection in the pop-up menu. Their callbacks get data from the handles structure and then plot it.

1.7 APPLICATIONS

- AI, Data Science, and Statistics.
- Mathematics and Optimization.
- Signal Processing.
- Image Processing and Computer Vision.
- Control Systems.
- Test and Measurement.
- RF and Mixed Signal.
- Wireless Communications.

CHAPTER-2

EXISTING METHOD

2.1 DIFFERENT TYPES OF IMAGE FOEGERY DETECTION

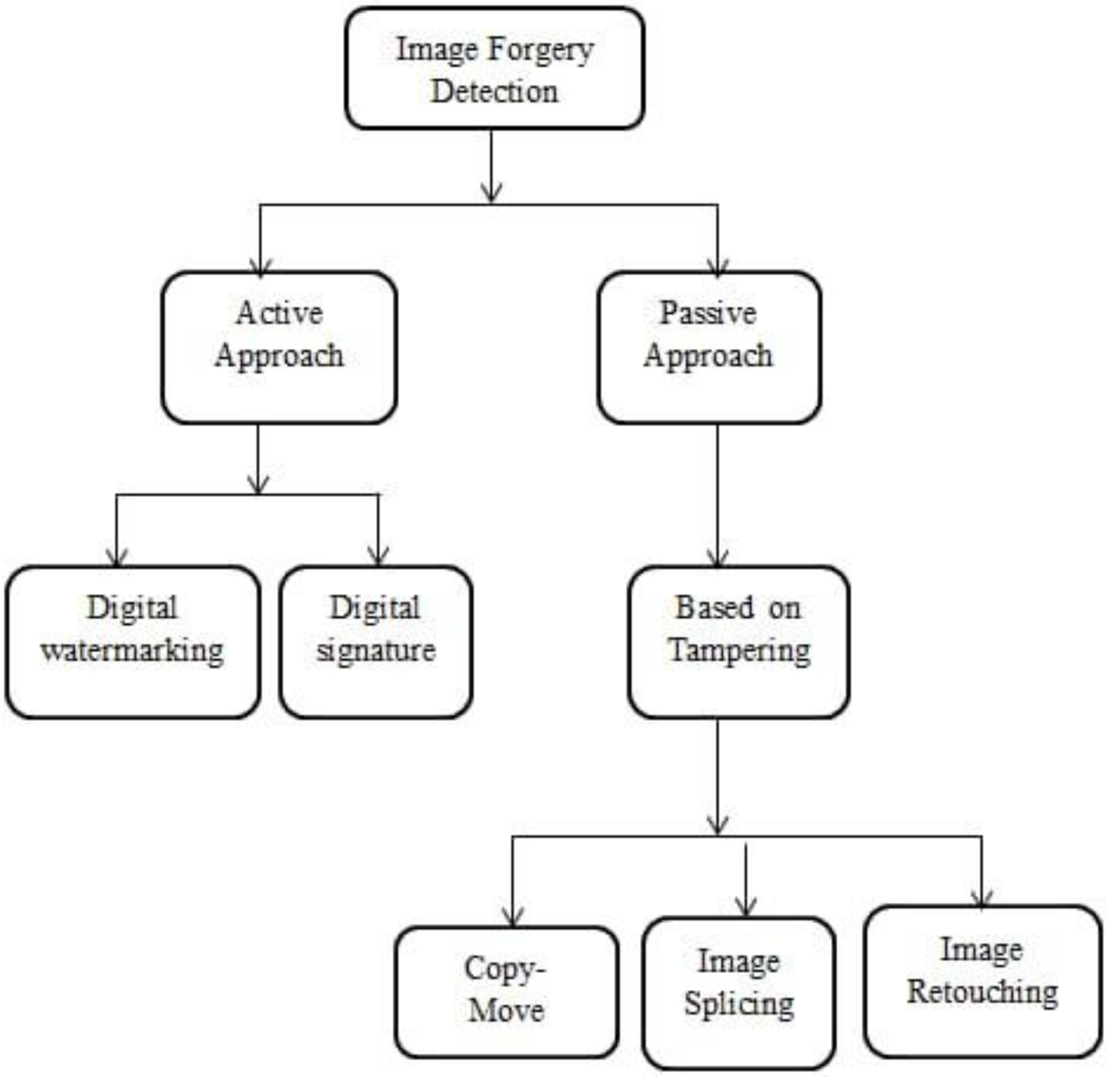


Fig.2.1 Types Of Image Forgery Detection

2.2 PASSIVE APPROACH

In this the identity of the client is checked and confirmed without requiring specific additional actions for the purpose of authentication. It is also behave as blind detection. In this technique there is no prior information related to Image.Assessing the originality and authenticity we used passivedetection technique on images, without any using of active technique like watermarking and digital Signature[3].

These are based on the assumptions which tell there are no clues of forged region on digital image and this may disturb the underlying image regularity of our surrounding sightimage that initiates new artifacts manufacturing in numerous types of anomalies.Forged region of image is also referred as anomaly of the image.

The techniques of passive image authentication chiefly classified into 5 types in the objects and light interaction with 3D. For capturing aimage light very important. Think about the twinkling of stars in the night, walking down in the garden to see the stars. Both types of images created by cloning together to make one image. This comes under Physical based forgery.

Geometric based technique in this technique, detection ofthe forgeries involved by measurements of objects at geometry level.

2.2.1 COPY MOVE FORGERY

It is one of the most common image tampering techniques, and it’s also one of the most difficult to spot because the cloned image is taken from the same image. A section of an image is copied and pasted to another part of the same picture in Copy-Move image forgery.

Copy-Move is a type of forgery in which a part of image is copied and then pasted on to another portion of the same image. The main intention of Copy-Move forgery is to hide some information from the original image. Since the copied area belongs to the same image, the properties of copied area like the color palette, noise components, dynamic range and the other properties too will be compatible with the rest of the image . So, the human eye usually has much more trouble detecting copy-move forgeries. Also forger may have used some sort of retouch orresample tools to the copied area so as it becomes even more difficult to detect copy-moved forgery. Retouching involves compressing the copied area, adding the noise to the copied area etc. and re-sampling may include scaling or rotating the image. It is a forged image, which is created by modifying is created by copying small portion from the same image and then pasted onto another area of same image. Original image and is used to create forgery [7].

PCA: Principal Component analysis,

DCT: Discrete Cosine transform,

DWT: Discrete Wavelet transform,

SVD: Singular Value decomposition,

SIFT: Scale Invariant Feature transform,

SURF: Speeded up Robust features.

Principle Component Analysis (PCA) is a candidate to extract the image features [9]. In the Dartmouth Computer Science Technical Report of 2004, Alin C Popescu and Hany Farid used PCA to automatically detect duplicated regions in a digital image. The technique works by first applying a principal component analysis to small fixed-size image blocks to yield a reduced dimension representation. The representation is robust to minor variations in the image due to additive noise or lossy compression. Duplicated regions are then detected by lexicographically sorting all of the image blocks. The efficiency of the proposed technique on credible forgeries, and its robustness and sensitivity to additive noise and lossy JPEG compression has been shown by the author.

The procedure to produce each feature vector is called principle component analysis in which values are obtained by using the theorems of covariance matrix, eigenvectors and linear basis for each image block with the initial conditions of zero-mean. Then a matrix S of block vectors quantized according to number of quantization bins to reduce the mirror variations created. These quantization coefficients are then sorted lexicographically and the duplicated regions has been detected by considering the offset of all pairs whose distances in S less than a specific threshold. To obtain the efficient results, a duplication map is defined by producing a zero image of the same size as original and assigning all pixels in a duplicated region to a unique grayscale value. With the dimension of the PCA reduced representation and total number of image pixels are Nt and N respectively, the algorithm has complexity of O(NtNLogN).

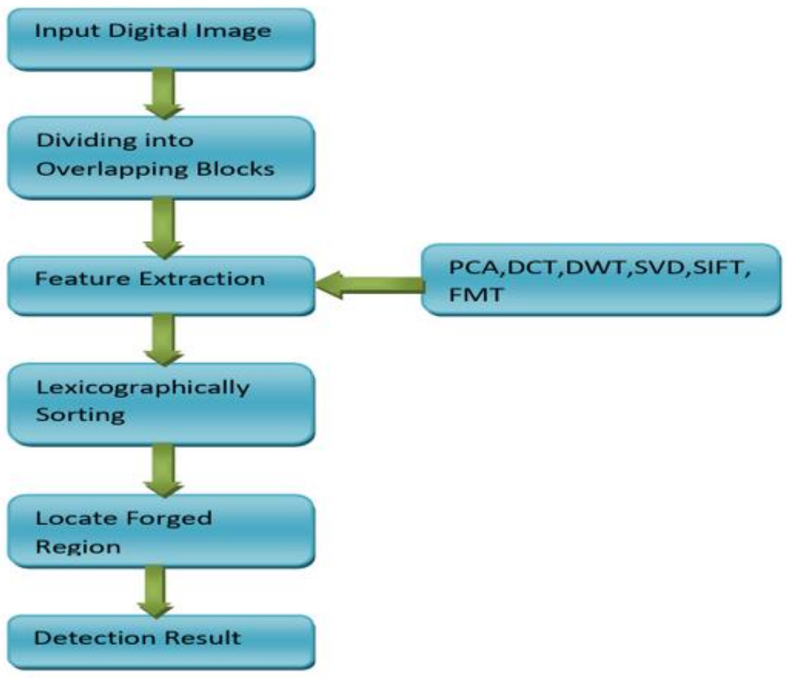

Input Digital Image
Dividing into Overlapping Blocks
Feature Extraction
PCA,DCT,DWT,SVD,SIFT, FMT
Lexicographically Sorting
Locate Forged Region
Detection Result

Fig. 2.2.1.1 Flow Chart Of CMFD Detection

2.2.2 DCT, DWT

Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) are popular techniques to transform an input image to the frequency domain before extracting the image features. Jessica Fridrich, et.al [8] used quantized Discrete Cosine Transform (DCT).The feature vectors are vectors of quantized DCT coefficients. The quality factor in JPEG compression determines the quantization step for DCT transform coefficients which is called the user-specified parameter Q. The mutual positions are considered in case of too many matching blocks having the same shift vector to define a specific block pair. For the color images, the algorithm requires a color to grayscale conversion. It takes MNlog2(MN) steps in exact match. Discrete Wavelet Transform (DWT) is always the potential candidate for research on CMFD. In fact, many proposed algorithms to detect Copy-Move regions using DWT coefficients. Nandini Singhal, et. al [4] proposed a system using DWT to detect pixel-based forgery. A forged image is taken as an input image. DWT is applied to the input image to yield LL1 sub-band. The LL1 sub-band is divided into sub-images. Then phase correlation is calculated. The offset between the copy-move regions is also calculated. The copy-move region is found out by pixel matching. Then MMO (Mathematical Morphological Operations) is applied to detect the result.

2.2.3 SIFT, SURF

Scale Invariant Features Transform (SIFT)presents a method for extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene in given image. The features are invariant to different scale and rotation, and provide robust matching across a large range of affine transformation, distortion, change in 3D viewpoint, addition of noise, and change in illumination. The features are highly distinctive and a single feature can be correctly matched with high probability against a large database of features of image and video. SURF is a scale and rotation invariant interest point detector and descriptor. It can be computed and compared much faster than other image features like SIFT and HOG.

When some object is copy-moved with the help of geometrical and illumination transform, it becomes difficult to detect that object. Speed up Robust Feature (SURF) and Scale Invariant Feature Transform (SIFT) are invariant with respect to geometrical and illumination transform. Ramesh Chand Pandey, et. Al (2014) [10] proposed a method to detect copy move forgery in an image based on passive forensic scheme. The proposed method used SURF and SIFT, which make it very fast and robust in detecting copy-moved regions. To achieve very fast speed in copy-move forgery detection, SURF image features are used to find the image key-points (interest points) and extract their64 dimensional descriptor which is used for rapid matching. The purposed system reads an image. Image key-points interest points are detected via SURF key-point detector. The descriptors for the key-points are computed using SURF key-point descriptor. The best ten matches are identified for every key-point. g2NN matching (g2NN-generalized 2 nearest neighbor) is applied. The dynamic thresholding step is performed. The matched key-point are joined using a line to represent the copied region.

It consists of two attacks: class-1 copy-move forgery and class 2 copy and creates a forgery. The methods deployed for Copy-Move forgery detections are block-based and key point based listed below .

So, the human eye usually has much more trouble detecting copy-move forgeries. Also forger may have used some sort of retouch or resample tools to the copied area so as it becomes even more difficult to detect copy-moved forgery. Retouching involves compressing the copied area, adding the noise to the copied area etc, and re-sampling may include scaling or rotating the image.

It is a forged image, which is created by modifying is created by copying small portion from the same image and then pasted onto another area of same image. is original image and is used to create forgery.

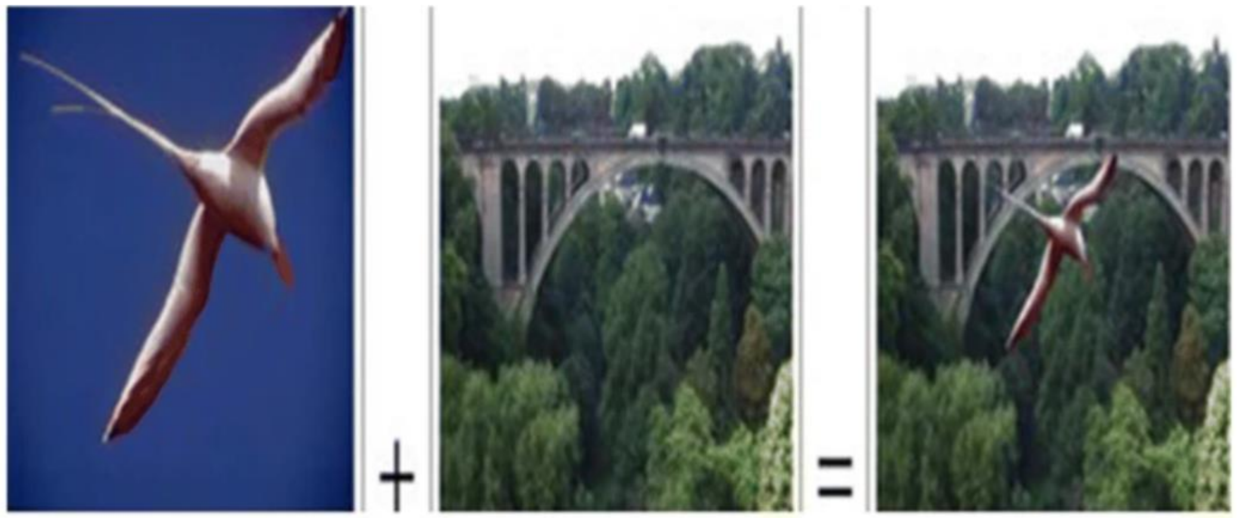


Fig 2.2.3   Example Of Copy Move Forgery

2.3 IMAGE RETOUCHING

Image retouching is one more type of image forgery tool which is most commonly used for commercial and aesthetic applications. Retouching operation is carried out mostly to enhance or reduce the image features. Retouching is also done to create a convincing composite of two images which may require rotation, resizing or stretching of one of the image. Retouching is the process of removing or inserting some given targets into a real photo or image to deliberately to produce an another image. It finds its use in many applications like image coding, journalism, art, film special effects production, virtual reality simulation etc.. Artists professionally retouch the images to restore old historical paintings, Malicious users may also use these manipulations to Tamper the digital images.

In Image Retouching, the images are less modified. It just enhances some features of the image. There are several subtypes of digital image retouching, mainly technical retouching and creative retouching**.**

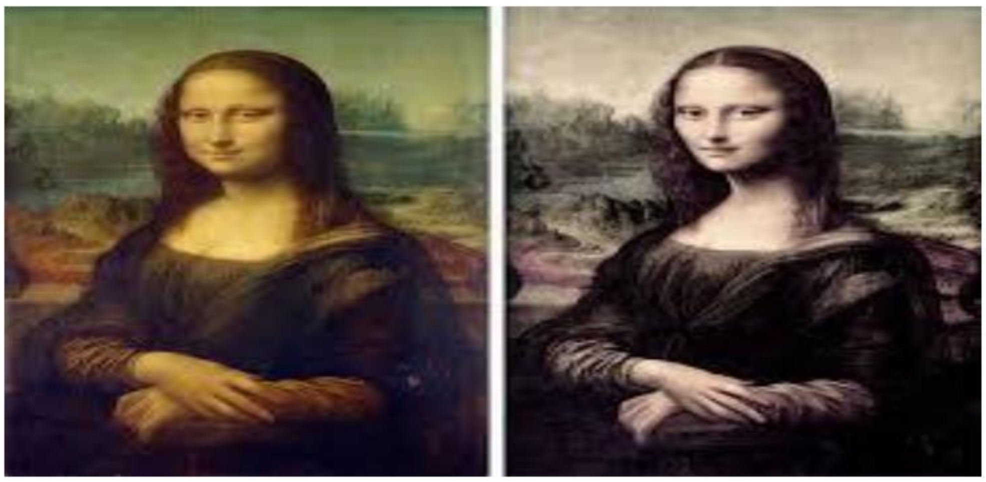
Image retouching is one more type of image forgery tool which is most commonly used for commercial and aesthetic applications. Retouching operation is carried out mostly to enhance or reduce the image features. Retouching is also done to create a convincing composite of two images which may require rotation, resizing or stretching of one of the image.

Retouching is the process of removing or inserting some given targets into a real photo or image to deliberately to produce an another image. It finds its use in many applications like image coding, journalism, art, film special effects production, virtual reality simulation etc. Artists professionally retouch the images to restore old historical paintings, Malicious users may also use these manipulations to Tamper the digital images.

Image retouching is also called Image In painting and is used in enhancing or reducing image features and it does not conform to the standards of morality. Image retouching detection is carried out by trying to find the blurring, enhancements, colour changes and illumination changes in the forged image.

Detection is easy if the original image is available however blind detection is a challenging task. For this type of forgery two type of modification is done either global or local. Local modification is done usually in copy move and in splicing forgery.

Contrast enhancement that is carried out in case of retouching is done at global level and for detection of tampering these are investigated. For illumination and changes in contrast global modification is carried out.



Fig(a) Original Image **Fig(b) Forged Image**

Fig.2.3 Example Of Image Retouching

CHAPTER-3

PROPOSED METHOD

3.1 INTRODUCTION

The advancements in digital image-editing software over the past decade have made image manipulation accessible to the masses. Consumers nowadays have unrestricted access to hundreds of advanced image editors, which also happen to be available for mobile devices, such that it has never been more convenient to morph and tamper images. The saturated market of application has forced developers to come up with novel ways to edit images, making the process of image forgery not only effortless and straightforward, but also accurate and imperceptible.

As a result, human eyes can no longer differentiate forged from original images; and, although the consequences of image tampering are beyond the scope of this discussion, it is undeniable that this may exacerbate the spread of misinformation and fake news. The act of image forgery can fall under either image manipulation such as the case of “copy-move”, or “splicing”

In the former type (*i.e.*, copy-move), a portion of an image is transposed into a different location within the same image in a way that cannot be (easily) recognized by the naked eye. In contrast to copy-move forgery, image splicing represents the act of copying contents from an image into a different image.

Since image forgery can be considered as a binary condition (*i.e.*, either authentic (original) or tampered (forged)), it can be automatically classified using the classification techniques of machine learning. In image forensics, the image is processed through two main detection methods: active and passive (also known as blind) . The active detection entails the use of additional information which is inserted into the image prior to distribution, such as in the case of digital water marking. In contrast, the passive detection employs statistical approaches to detect alterations in the features of an image. Corresponding to the increase in image forgery, forgery detection methods have thrived over the years, with more sophisticated algorithms being proposed every year.

3.2 IMAGE SPLICING

Image splicing is a forgery manner to copy and paste regions within separate image sources, usually carried out by digital image editing tools such as Photoshop. It is often used as an initial step of photomontage, which is very popular in digital image content editing. The splicing tampered image could be used in news reports, photography contest, key proof in the academic papers, and so on, which could bring certain negative influences [4].
As a result, it is an important issue to develop reliable splicing detection methods.

Image splicing is an image editing method to copy a part of an image and paste it onto another image, and it is commonly followed by postprocessing such as local/global blurring, compression, and resizing.

To detect this kind of forgery, the image rich models, a feature set successfully used in the steganalysis is evaluated on the splicing image dataset at first, and the dominant sub model is selected as the first kind of feature.

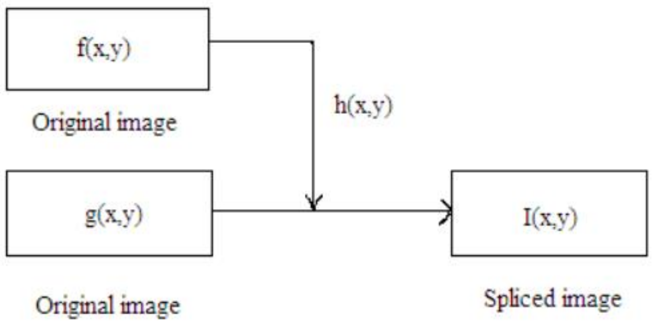


Fig.3.2.1 The steps of image splicing,

f(x, y) and g(x, y) are original images, h(x, y) is a part of f(x, y) which is insert into g(x, y) and generate spliced image I(x, y). Perhaps f(x, y) and g(x, y) is the same image.

The selected feature and the DCT Markov features are used together to detect splicing forgery in the chroma channel, which is convinced effective in splicing detection. The experimental results indicate that the proposed method can detect splicing forgeries with lower error rate compared to the previous literature.

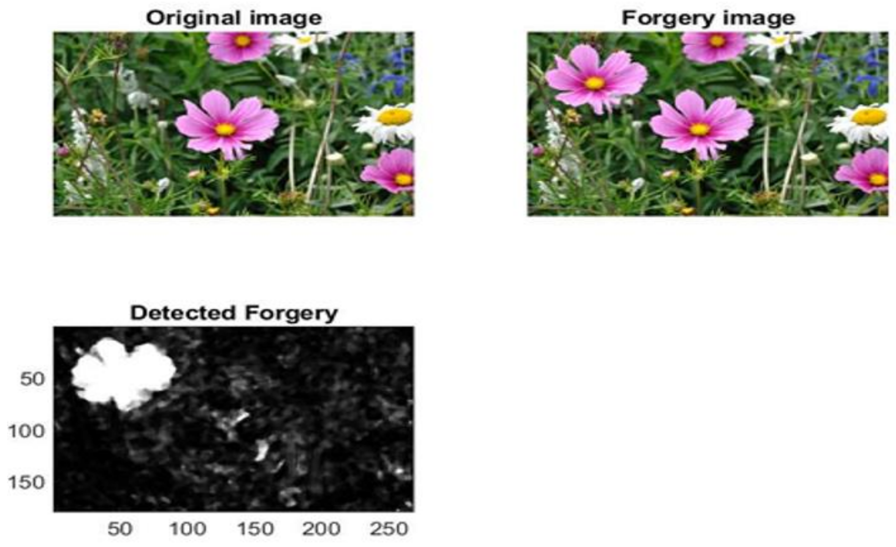


Fig.3.2.2 Example Of Image Splicing Forgery Detection

3.3 RELATED WORK

Over the years, many passive algorithms have been proposed for image splicing detection. These algorithms employ different techniques for features extraction, including the "Local Binary Pattern" (LBP), Markov, Transform model (wavelet transform, Discrete Cosine Transform (DCT) and deep learning. In the first type, the LBP algorithm is applied as a feature extraction from the spliced images, by Zhang et al. This image splicing detection was based on the "Discrete Cosine Transform" (DCT) and LBP [5]. The DCT is applied to each block of the input image, and then features were extracted by the LBP approach. The final feature vector classified by using the SVM classifier. Alahmadi et al. proposed an algorithm for passive splicing detection which is based on DCT and “local binary pattern” (LBP). This algorithm relies on initial conversion of the input image from RGB into YCbCr, then the LBP output is generated by dividing the converted image’s chrominance channel into overlapping blocks. The LBP output is converted into the DCT frequency domain to enable the use of the DCT coefficients as a feature vector. The feature vectors are then used with the SVM classifier, which is responsible for deciding whether an image is forged or authentic. This method reportedly achieved accuracies of 97%, 97.5% and 96.6% when tested with the three dataset “CASIA v1.0 & v2.0” and “COLUMBIA” respectively. In the same approach, In Han et al. , proposed a feature extraction based on “Markov features” to detect the spliced images based on the maximum value of pixels in the DCT domain. The high numbers of extracted features were reduced by using the even-odd “Markov algorithm”. However, the limitation of Markov models is the high complexity and time consumption. Accordingly, Jaiprakash et al [6].

proposed a “passive” forgery detection technique where image features are extracted from the “discrete cosine transforms” (DCT) and “discrete wavelet transform” (DWT) domains. Their approach employed the ensemble classifier for both training and testing to discriminate between forged and authentic images. The algorithm operates under the Cb + Cr of the YCbCr color space, and the authors showed that the features obtained from these channels demonstrated better performance compared to features extracted by using individual Cb and Cr channels. The algorithm has reportedly achieved classification accuracies of 91% and 96% for “CASIA v1.0” and “CASIA v2.0” datasets, respectively. Subramaniam et al. utilized a set of “conformable focus measures” (CFMs) and “focus measure operators” (FMOs) to acquire “redundant discrete wavelet transform” (RDWT) coefficients that were subsequently used to improve the detection of the proposed splicing detection algorithm. Since image splicing causes disfigurement in the contents and features of an image, blurring is usually employed to flush the boundaries of the spliced region inside the image.

Even though this may reduce the artifacts generated by splicing, the blurring information can be exploited to detect forgeries. Both CFM and FMO were utilized to measure the amount of blurring that exists in the boundaries of the spliced region in the aforementioned algorithm. The 24-D feature vector algorithm was tested with two public datasets “IFS-TC” and “CASIA TIDE V2” and has reportedly achieved accuracy rates of 98.30% for the Cb channel from the former dataset and 98.60% for the Cb channel from the latter dataset. With such accurate classification performance, this method outperforms other image splicing detection methods. Moreover, the third type of feature extraction for detection the spliced images is suggested by El-Latif et al. This approach presented a deep learning algorithm and wavelet transform for detecting the spliced image. The final features are classified by SVM classifier. Two publicly image splicing datasets (CASIA v1.0 and CASIA v2.0) were used to evaluated the method. The large number of features with high complexity of calculations are the main limitations.

Wang et al. proposed an approach that employs the "convolutional neural networks" (CNN) with a novel strategy to dynamically adjust the weights of features. They utilized three feature types, namely YCbCr, edge and "photo response non-uniformity" (PRNU) features, to discriminate original from spliced images. Those features were combined in accordance to a predefined weight combination strategy, where the weights are dynamically adjusted throughout the training process of the CNN until an ideal ratio is acquired. The authors claimed that their method outperforms similar methods that also utilize CNN, in addition to the fact that their CNN has significantly less depth than the compared methods, which overall counts as an advantage.

3.4 PROPOSED ENHANCEMENT METHOD

In the present study, we propose a texture feature-based algorithm in which four features (*i.e.*, FrEp, LBP, skewness and kurtosis) are extracted from a YCbCr-converted image. These features are then combined together to obtain a feature vector. The resulting 4D feature vector is subsequently used for the SVM classifier to determine whether an image is spliced or authentic. The algorithm comprises three main steps: first, in the pre-processing step, the RGB image is transformed into the YCbCr format; second, in the feature extraction step, image features (*i.e.*, FrEp, LBP, skewness and kurtosis) are acquired from the YCbCr-converted image and are combined together to obtain a feature vector; lastly, in the classification step, SVM is used with the obtained feature vector to discriminate spliced from authentic images.

The classification accuracy of the proposed method is further enhanced by employing a combination of texture features and reducing the total feature dimension.

3.5 PREPROCESSING

In this step, the input image is converted from its original RGB color space into the YCbCr color space where Y symbolizes the luminance, and Cb and Cr characterize the chrominance color. Compared to the latter two channels, the Y channel holds the most information; thus, any changes to this channel will lead to prominent changes to the image, which can be recognized by the naked eye. In contrast, the information held by the Cb and Cr channels does not visibly a effect the image, and therefore, any changes to these channels are more difficult to spot. In light of the above, the proposed method utilizes the chrominance channel for features extraction.

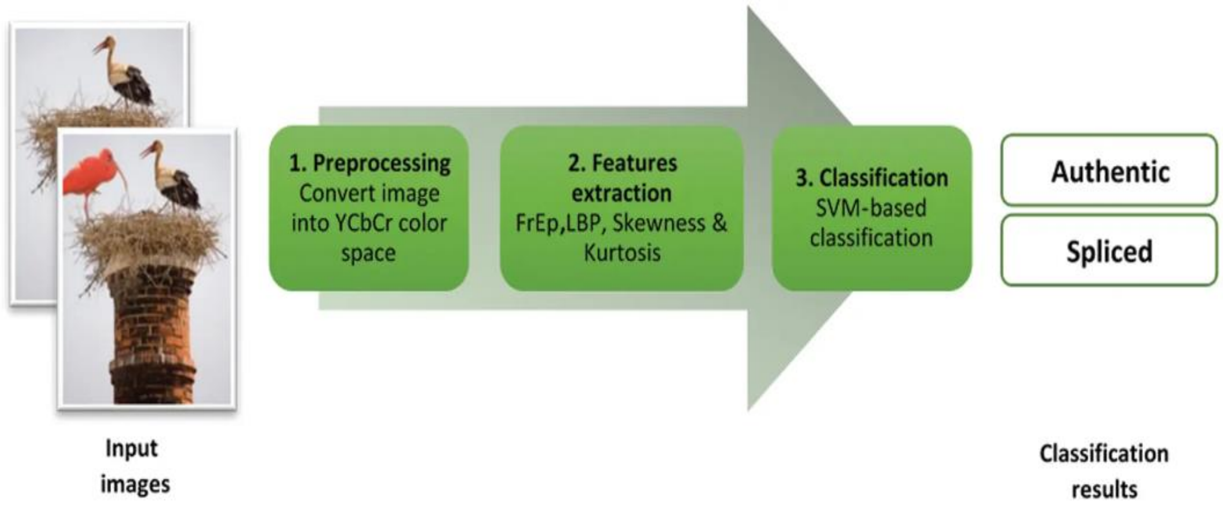


Fig.3.4 A diagram depicting the proposed method

3.6 FEATURE EXTRACTION

Since the process of image splicing involves transformations to the image which include: translation, rotation and scaling, the proposed method extracts the following features from the Cb and Cr channels: FrEp, LBP, skewness, and kurtosis. These features have been chosen in the present study because they are amongst the most used features reported in the literature, and they give good representation of the texture of an image. LBP is an effective image descriptor to define the patterns of the local texture in images by capturing the local spatial patterns and the gray scale contrast in an image. It is extensively applied in the different image processing applications .

The most important component of image splicing detection is to have sensitive features to any alterations due to tampering. The of textural features distributions offering a statistical basis for separation between authentic and spliced images. Moreover, three statistical features such as FrEp, skewness and kurtosis are used extracted image features from each Cb, and Cr image in order to identify the differences between the spliced or authentic images. The state of the image texture is of great importance in splicing detection since all alterations done to the image are reflected by altered image texture .

3.6.1 Fractal Entropy (FrEp)

Entropy is a statistical measure of randomness, which can be used to gauge the texture of an image. It calculates the brightness entropy of each pixel of the image, and therefore, it is defined as

$$E=-\sum nm=1Gm*\log2(Gm)$$

where G(G1, G2, …, Gn) represents normalized histogram counts returned from the histogram of the input image. The entropy as a texture descriptor mostly provides randomness of image pixel with its local neighbor hood. Ubriaco formalized the definition of Shannon fractional entropy as follows:

where p is the pixel probability of the image, and α is the fractional power of entropy (the order of entropy). Valério et al. generalized as follows:

$$S\alpha Valerio=\sum n([-\ln(pn)]\alpha\Gamma(\alpha+1)p\alpha n)pn$$

The local fractional calculus is also used to define a modified fractal entropy as follows

$$S\alpha fractal=\sum n([-\ln(pn)]\alpha n\Gamma(\alpha n+1)p\alpha n)pn$$

In this study, we introduce more modification on fractional entropy using Re'nyi entropy. Since Re'nyi entropy satisfies the following relation

$$SR\alpha=\ln[1+(1-\alpha)S\alpha Ubriaco]1-\alpha,$$

By substituting the fractal Re'nyi entropy of (3) into (4), we get the proposed FrEp as follows:

$$FrEp=\ln[1+(1-\alpha)S\alpha fractal]1-\alpha$$

The logic behind using Fractal Entropy as a texture feature extraction is that the entropy and the fractal dimension are both considered as spatial complexity measures. For this reason, the fractal entropy has the ability to extract the texture information contained in the input image efficiently. The main advantage of FrEp lies in their ability to accurately describe the information contained in the image features, which makes them an efficient feature extraction algorithm. In the proposed FrEp feature extraction model, the key parameter is α, where the performance of the FrEp basis function of the α power is utilized to enhance the intensity value of the pixels of the image, which might influence the accuracy of the detection process of image splicing. The optimal value of α has be chosen experimentally equal to 0.5.

3.6.2 LBP Based Features

LBP, like proposed FrEp, describes the texture state of an image. In LBP, pixel values are transformed into a binary number using thresholding[10]. This is done by considering the binary value in a clockwise fashion, beginning with the top-left neighbor. LBP can be defined as follows:

$$LBPpq(xm,ym)=\sum P-1m=0F(Jm-Jct)2m$$

where Jm represents the m neighborhood pixel intensity value, and Jct is the central pixel value, p is the sampling points, and q is the circle radius.

$$F(m)=\{1 \text{ if } m\geq 0\text{ , }0 \text{ if } m<0$$

The image texture extracted by the LBP is characterized by the distribution of pixel values in a neighborhood, where each pixel is modified according to thresholding function F(m).

3.6.3 SKEWNESS

Skewness is a statistical quantity of asymmetry distribution of a variable, and can be defined as:

Skewness=1σ3[1n∑ni=1(xi−μ3)]

where σ represents the standard deviation, μ denotes the mean of an image, and n is the number of pixels.

3.6.4 KURTOSIS

Kurtosis is a measure used to describe the form (peakedness or flatness) of a probability distribution. The formula for kurtosis is as follows:

Kurtosis=1σ4[1n∑ni=1(xi−μ4)]

3.7 PROPOSED ALGORITHM

These features have been chosen due to its ability to show the significant detail of the image. These four features highlighted the texture detail of the internal statistics of forged parts. The proposed method is summarizing as follows:

(1) Convert the image color space into YCbCr color space.

(2) Extract the Cb and the Cr images.

(3) Split the input image into non-overlapping image blocks of size of 3 ×× 3 pixels.

(4) Extract the four proposed texture features (FrEp, LBP based features, Skewness, and Kurtosis) each block from Cb and Cr images.

(5) Save the extracted features vector as the final texture features for all Cb, and Cr image.

(6) Apply the SVM to classify the input image into “authentic” or “spliced forged image”.

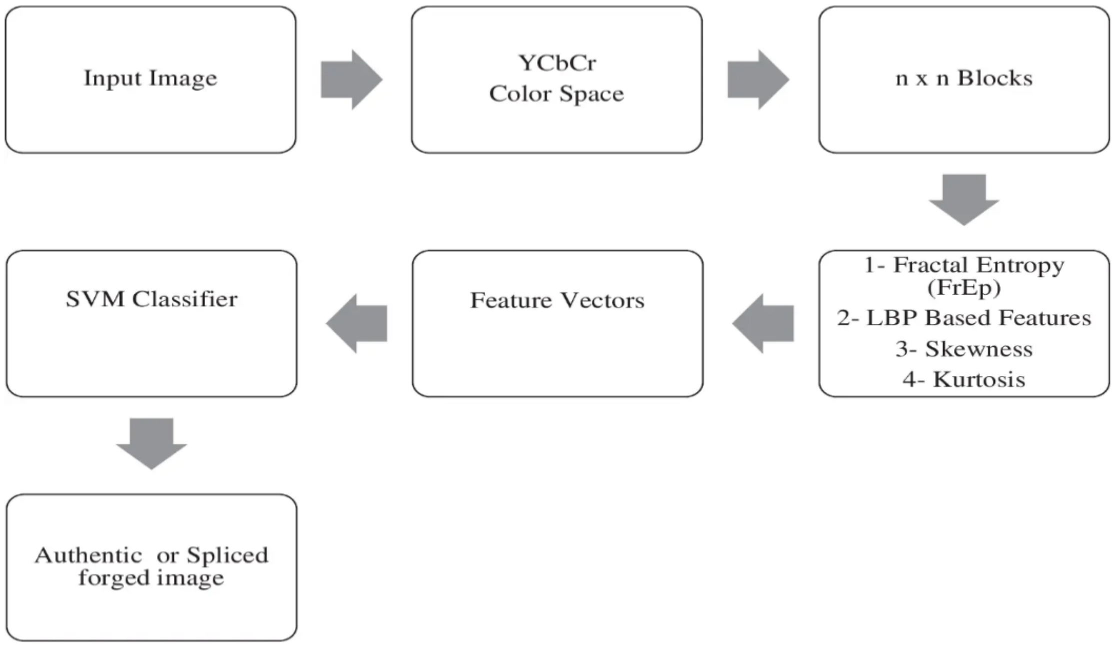


Fig.3.7 Proposed Algorithm

CHAPTER-4

CODING FOR IMAGE FORGERY DETECTION

PROGRAM

```
clc
clear all
close all
filename = {'yash .jpg','yash tampered.jpg'};
% dimensione of statistics
Nb = [2, 8];
% number of cumulated bloks
Ns = 1;
bayer = [0, 1;
         1, 0]
for i = 1:2:8
    im_true = imread(filename{i});
    im = imread(filename{i+1});
    for j = 1:2
        [map, stat] = CFAloc(im, bayer, Nb(j),Ns);
       [h w] = size(map);
       %    NaN and Inf management
       stat(isnan(stat)) = 1;
       data = log(stat(:));
       data = data(not(isinf(data)|isnan(data)));
       % square root rule for bins
       n_bins = round(sqrt(length(data)));
       % plot result
       figure
       subplot(2,2,1), imshow(im_true), title('Original image');
       subplot(2,2,2), imshow(im), title('Forgery image');
       subplot(2,2,3), imagesc(map), colormap('gray'),axis equal, axis([1 w 1 h]), title('Detected
Forgery');
%      subplot(2,2,4), hist(data, n_bin s), title('Histogram of the proposed feature');
       display('Press any key to continue...')
       pause
    end
end
```

CHAPTER-5

SIMULATION RESULTS

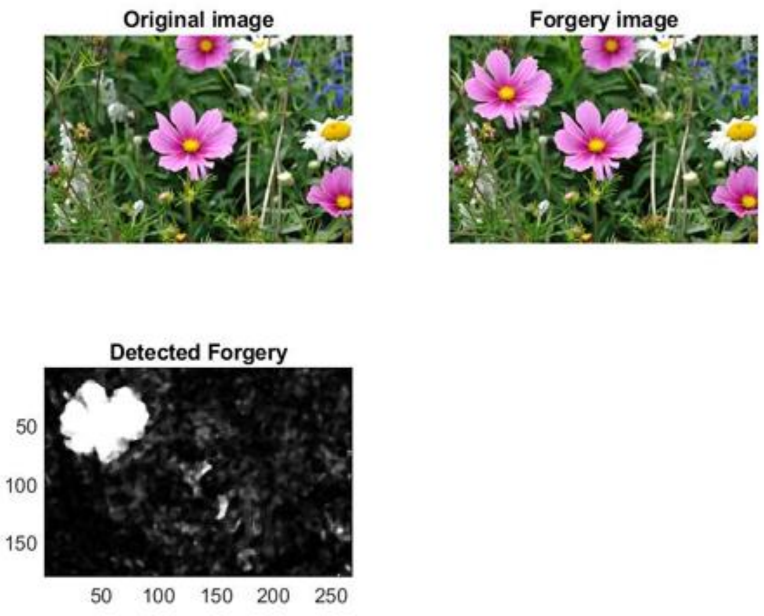


Fig.5.1 Detected Output 1

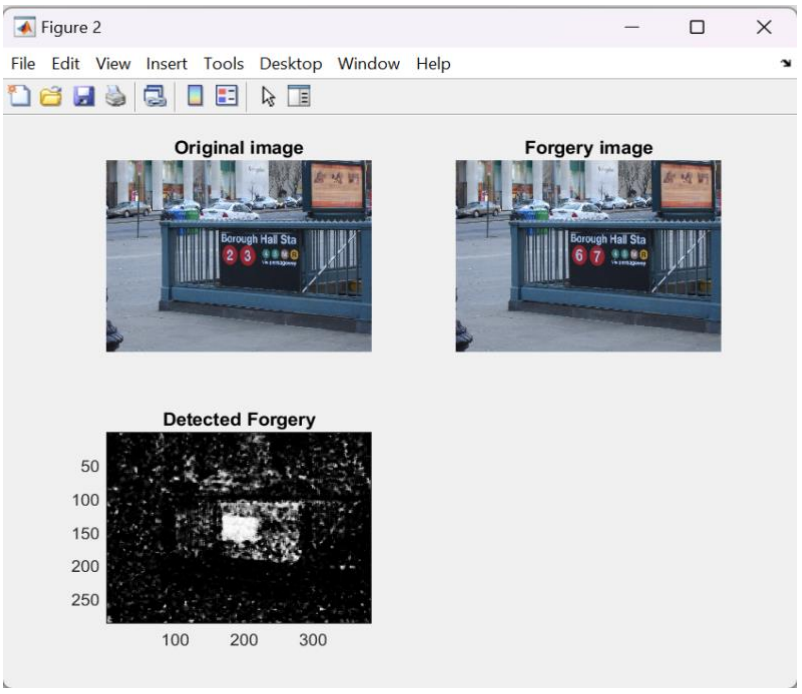


Fig.5.2 Detected Output 2

CHAPTER-6

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

In this work, we proposed an automatic tool to discriminate between spliced and authentic images using the SVM classifier. We have extracted the texture features using four features extraction stages namely, Fractal Entropy (FrEp), local binary patterns (LBP), Skewness, and Kurtosis to get cues of any type of manipulation on images in order to enhance the classification performance of the SVM classifier. Experimental validation on “CASIA v1.0 & v2.0” image datasets shows that the proposed approach gives good detection accuracy to identify the tampered images with reasonable feature extraction time.

6.2 FUTURE SCOPE

Proposed model gives higher detection accuracy than that of those methods with shorter feature extraction time. The proposed work has demonstrated striking accuracy rates of 96% and 98% when tested with the very versatile and comprehensive “CASIA v1.0 & v2.0” datasets respectively. These rates are superior to some of the recent state-of-the-art splicing detection methods. The experimental findings showed that the proposed image splicing detection method helps for the detection splicing attack in images using image texture features with proposed fractal entropy. In future splicing detection works, one could consider locating the spliced objects within a forged image.

6.3 REFERENCES

1. S. Sadeghi, S. Dadkhah, H. A. Jalab, G. Mazzola and D. Uliyan, “State of the art in passi ve digital image forgery detection: Copy-move image forgery,” Pattern Analysis and Ap plications, vol. 21, no. 2, pp. 291–306, 2018.
2. A. S. Kapse, Y. Gorde, R. Rane and S. Yewtkar, “Digital image security using digital wa termarking,” International Research Journal of Engineering and Technology, vol. 5, no. 3 , pp. 163–166, 2018.
3. I. E. A. El-Latif, A. Taha and H. Zayed, “A passive approach for detecting iImage splici ng using deep learning and Haar wavelet transform,” International Journal of Computer Network & Information Security, vol. 11, no. 5, pp. 1–9, 2019.
4. Y. Zhang, C. Zkao, Y. Pi, S. Li and S. Wang, “Image-splicing forgery detection based on local binary patterns of DCT coefficients,” Security and Communication Networks, vol . 8, no. 14, pp. 2386–2395, 2015.
5. A. Alahmadi, M. Hussain, H. Aboalsamh, G. Muhammad and G. Bebis, “Splicing image forgery detection based on DCT and LBP,” in 2013 IEEE Global Conf. on Signal and In formation Processing, 2013, Austin, TX, USA, IEEE, pp. 253–256, 2013.
6. J. Han, T. H. Park, Y. Moon and K. Eom, “Efficient Markov feature extraction method f or image splicing detection using maximization and threshold expansion,” Journal of Ele ctronic Imaging, vol. 25, no. 2, pp. 1–8, 201
7. S. P. Jaiprakash, M. B. Desai, C. S. Prakash, V. H. Mistry and K. Radadiya, “Low dimen sional DCT and DWT feature based model for detection of image splicing and copy-mov e forgery,” Multimedia Tools and Applications, vol. 79, no. 39, pp. 29977–30005, 2020.
8. J. Wang, Q. Ni, G. Liu, X. Luo and S. Jha, “Image splicing detection based on convoluti onal neural network with weight combination strategy,” Journal of Information Security and Applications, vol. 54, no. 102523, pp. 1–8, 2020.
9. Y. Zhang, J. Goh, L. Win and V. Thing, “Image region forgery detection: A deep learnin g approach,” in Proc. of the Singapore Cyber-Security Conf. 2016, Singapore, vol. 14, p p. 1–11, 2016.
10. H. A. Jalab, H. Omer, A. Hasan and N. Tawfiq, “Combination of LBP and face geometri c features for gender classification from face images,” in 2019 9th IEEE Int. Conf. on Co ntrol System, Computing and Engineering, 2019, Penang, Malaysia, IEEE, pp. 158–161, 2019.