ABSTRACT

GENDER DETECTION FROM HAND SIGNATURES USING MYRIO FPGA

By

Aniket P. Ratnakar

August 2015

The goal of this project is to use pattern recognition techniques for the detection of a person's gender, using the hand signature of the person. The gender detection algorithm utilizes image processing methodologies to identify and evaluate certain parameters, such as the number of intersection points and amount curvature in the trace of the hand signature. The image of the signature is captured in real time from a webcam using LabVIEW (Laboratory Virtual Instrument Engineering Workshop) and the MyRIO FPGA (Field Programmable Gate Array) platform from National Instruments. It is subsequently processed in MatLab (Matrix Laboratory) using a combination of various filters. The system can be used to improve security of activities or transactions where a hand signature is required.

GENDER DETECTION FROM HAND SIGNATURES USING MYRIO FPGA

A PROJECT REPORT

Presented to the Department of Electrical Engineering

California State University, Long Beach

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

Committee Members:

Anastasios Chassiakos, Ph.D. (Chair)
Dr. I-Hung Khoo.
James Ary, Ph.D.

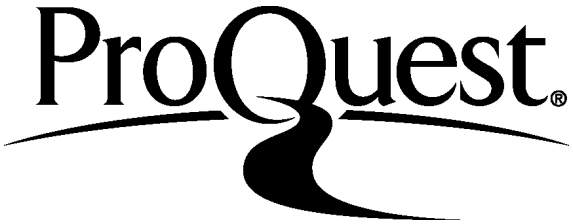
College Designee:

Antonella Sciortino, Ph.D.

By Aniket Ratnakar

B.E., 2013, Birla Vishvakarma Mahavidyalaya, India

August 2015



ACKNOWLEDGEMENTS

Working on a project is unique educating experience. The vast multitude of practical and technical exposure gained is available to us. This project is an illustration of our efforts to materialize “Gender Detection from Hand Signatures Using MyRIO FPGA.”

While submitting this report, it is relevant to avail ourselves of any opportunity to express our gratitude towards all those who guided and helped us in activating our work. We are grateful to the wholehearted support and contribution from number of individual leading towards success.

We have been very fortunate to get every possible aid and encouragement from our project guide, Dr. Anastasios Chassiakos. Lastly, I want to thank all those who helped us indirectly in successful completion of the project.

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF TABLES

TABLE Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER 1

INTRODUCTION

Exordium

The topic of gender detection has been the center of prime research topics since 20th century. It is proven that every person has a unique characteristic of producing hand signatures. Signatures have mainly influenced the legalization of government documents or documents that hold importance. It is almost impossible to mimic any person's signature as the pressure and curvature of strokes are purely unique to a person. Nowadays the hand signatures are converted into digital signatures to add an enhanced level of security to any important document. The theory of hand signatures is closely related to calligraphy. The only difference between them is calligraphy has a touch of artistic gesture while signatures are more security oriented.

Gender detection has various faces, with decision factors based on hand signatures, facial features, speech recognition and many more. The core of the algorithm discussed here lies in how pattern recognition works. The project mainly consists of implementing computer vision, pattern recognition, image reproduction, image segmentation and image filtering. The existing systems have large power requirements. Moreover they are hard wired and cannot be altered even though improved techniques get invented. Here comes Field Programmable Gate Arrays (FPGA), which are capable of taking shape of any hardware device. They can be remotely programmed and hence the hardware is retained, which ultimately reduces the electronic waste. The whole system here is a combination of MatLab running on a

computer, LabVIEW interface and the MyRIO FPGA. We implemented this system on National Instrument's MyRIO FPGA kit as this kit can be used in a standalone mode and can be useful to operate the system for real time application.

The gender detection algorithm is currently 70% accurate, which is a good number provided it requires only 20 images to be captured while other algorithms require thousands of images to be rendered. The whole detection takes place within 2.9 seconds. An external webcam is used to grab real time images, which adds to the plasticity of the system. The preprocessing block removes 98% of the noise from the picture, while the processing inside MatLab removes remaining noise giving ultra clean image for detection.

CHAPTER 2

LITERATURE SURVEY

An Overview of Gender Recognition

Signature recognition is called as a behavioral biometric. It is basically defined by two different ways: a) Static: In this method a signature is obtained by scanning a document using scanner or a camera, and the system recognizes the shape of the signature. b) Dynamic: In this method a signature is obtained using a tablet or a digitizer touch screen, other possibility is through a pen stylus on a capacitive screen. Amongst all the most popular technique is pattern recognition, which is discussed in this paper. The signatures are closely related to the personality of the person, it is exact image of what that person is thinking and hence a lot of information can be extracted from it.

History

Primary signature recognition system was developed during 1965, other dynamic recognition research continued in the 1970s which focused use of static characteristics rather than the dynamic one. The dynamic acquisition took place more from 1977 when the touch sensitive technologies became prevalent. During 1977, a patent was awarded that gave personal identification from dynamic pressure.

CHAPTER 3

HARDWARE DESCRIPTION

Hardware Specifications

NI MyRIO is an educational FPGA with advanced application support specifically for engineering modelling with major advantage of simulating real time systems. It is stacked up with latest Zynq System on chip technology from Xilinx, with a powerful dual-core ARM® Cortex™-A9 processor and an FPGA that has 28,000 programmable logic cells, 10 analog inputs, 6 analog outputs, 40 lines of digital input/output (DIO), and audio I/O channels. The MyRIO kit is a perfect fit for students doing academic projects. It comes with an onboard WiFi module, 3 axis accelerometer, and four programmable LEDs (Light Emitting Diode) in an enclosed box for circuit protection. MyRIO is customizable with the NI LabVIEW FPGA Module.

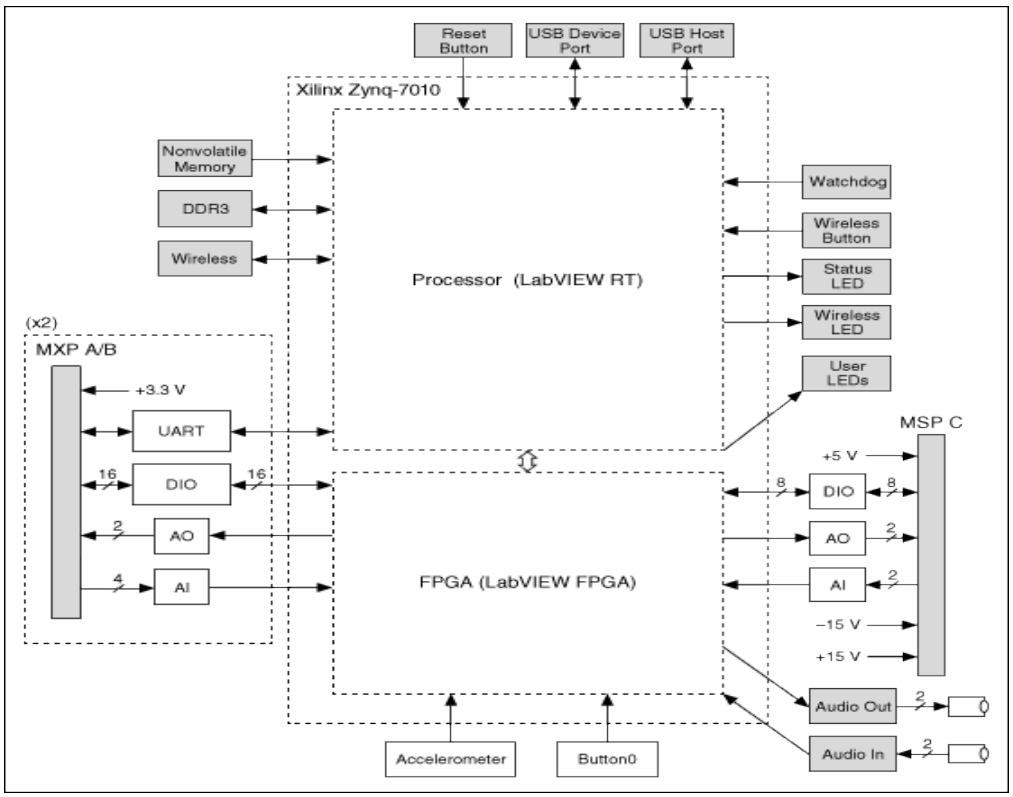


FIGURE 1. Block diagram of the kit.

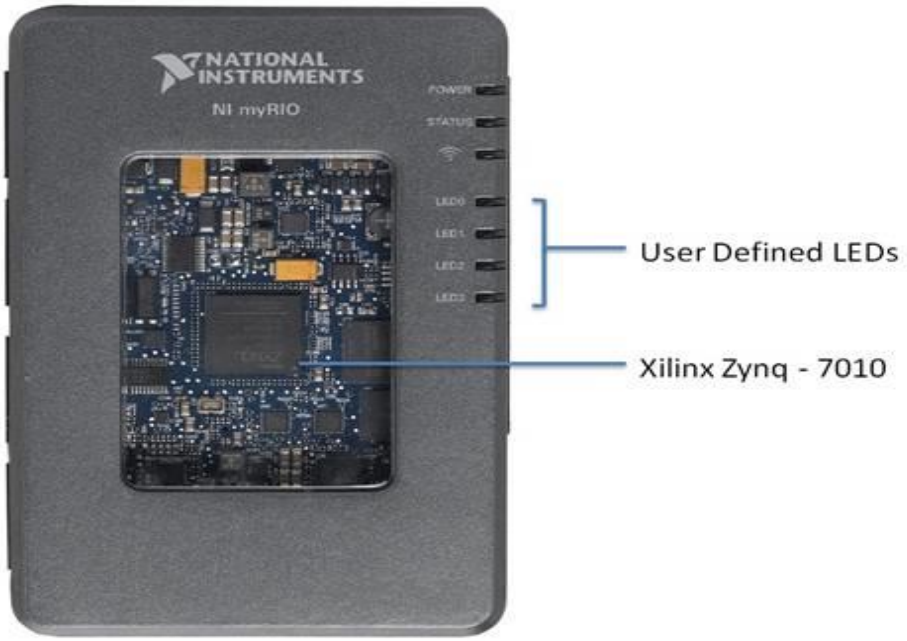


FIGURE 2. Top view of the kit.

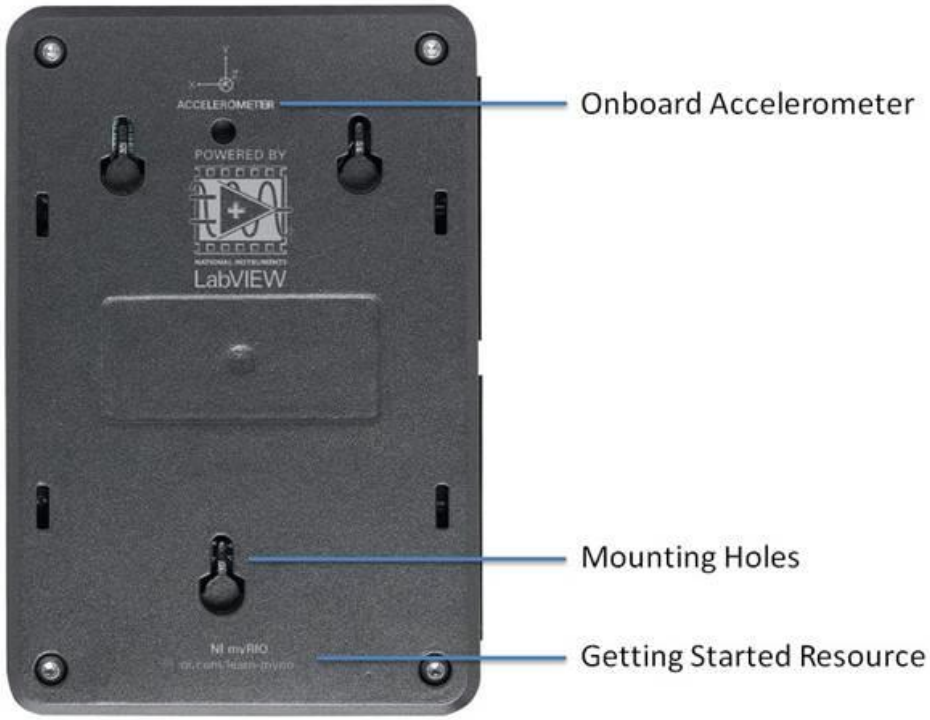


FIGURE 3. Back view of the kit.

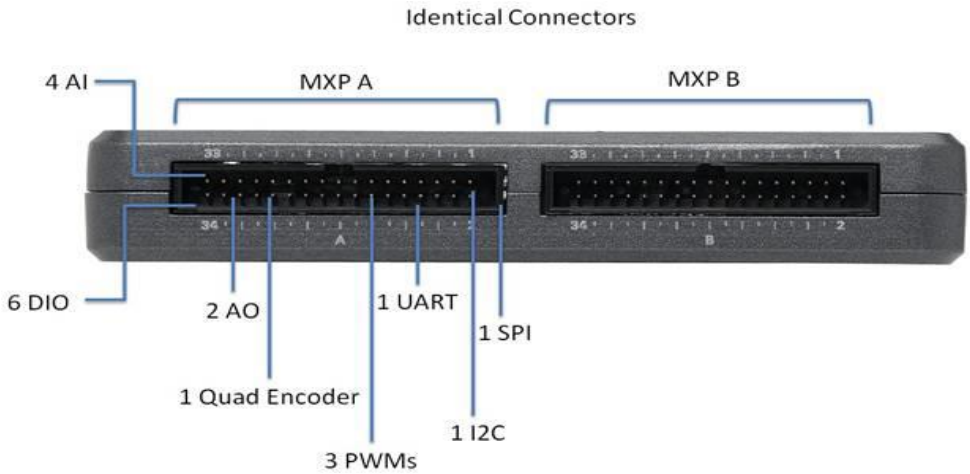


FIGURE 4. The default I/O configuration.

FIGURE 5. Extra ports for analog and digital interface.

CHAPTER 4

SOFTWARE DESCRIPTION

LabVIEW

Acronym for Laboratory Virtual Instrumentation Engineering Workbench is a platform that allows user to design real time systems with the help of graphical module and graphical language. The Graphical language is called "G," originally released for the Apple Macintosh in 1986, it is used for industrial automation and instrument control.

Dataflow Programming

LabVIEW uses a dataflow structured programming language which is called G language. Execution of the program is dependent on the structure of a graphical block diagram (the LV-source code) which is connected through different function nodes by a user. The wires propagate variable data and the block executes as the data is available. The G language has an excellent capability of parallel programming. The multi-processing and multi-threading functionality of the LabVIEW is implemented by scheduler which is built in the software. Programmer with knowledge would believe LabVIEW susceptible to race conditions. But this problem comes during data-flow programming. The above mentioned dataflow method completely defines the execution flow. The execution flow is well defined just like C, visual Basic and Python. Also, LabVIEW does not require variables to be type-defined; wire type is automatically defined by the supplying node. LabVIEW allows different types of data just like polymorphism through the wires.

Graphical Programming

LabVIEW contains a user interface which is called front panel and it can be used during the development cycle. LabVIEW is made up of programs called Virtual Instruments. Each VI is made up of three sections: Block diagram, Connector pane and the front panel. Some VI maybe connected through sub VIs (Virtual Instruments) which are connected by programming language. These sub VIs are used for importing and exporting data or the information. A front panel can be used as a programming language. This graphical approach provides more insight to understand for non-programmers. This interface is like lab equipment, which is easy to represent. Inbuilt examples and documentation in LabVIEW help build tools easily. This is good on one side but on the other a certain expertise is needed to understand G programming language. When it comes to complex algorithms it is necessary that a programmer has necessary knowledge of special LabVIEW syntax and structure of its memory management. Recent LabVIEW development systems offer functionality of building a stand-alone program. Furthermore, it is possible to create distributed applications which communicate by a client/server scheme, so it is easier to implement due to the inherently parallel nature of G-code.

Benefits

The benefits of LabVIEW are given as, LabVIEW supports platform independent hardware access to numerous data acquiring and instrumentation modules. LabVIEW packages include the libraries with a large number of functions for data acquisition, signal conditioning, generation, mathematics, analysis, statistics and numerous graphical interface elements. The environment provided by LabVIEW is supported by various operating systems like Windows, Mac and Linux. It is capable of deploying G–code in various targets including devices like Phar Lap OS based Lab

VIEW real-time controllers, Field Point modules and into FPGAs. It also includes abstraction layers which offer standard software interfaces to communicate with hardware devices.

NI Vision Development Module

National Instruments Vision Development Module is supported by LabVIEW and used to develop and deploy machine vision applications. It also supports various cameras and has the ability to acquire the video and images. Acquired images are processed by using many functions which are inbuilt in the LabVIEW software. Popular image processing operations performed using LabVIEW include checking for presence of the image, identifying objects, measuring size, extracting image details.

The various advantages of the module are:

Capturing and processing with a wide range of cameras to reduce the processing time and development cost, as well as reuse existing code when changing hardware. Example—Same software package is used in Linux and Windows for development of different types of FPGAs or multicore processor devices such as NI MyRIO kit.

Many algorithms are present in the Vision Development Module to meet any vision application challenge. Example—Module has built in algorithm for automatic brightness and contrast of the image.

Other than image processing, vision development module can also be used to communicate with other devices using a range of I/O options and protocols including Serial RS232, TCP/IP, and Ethernet.

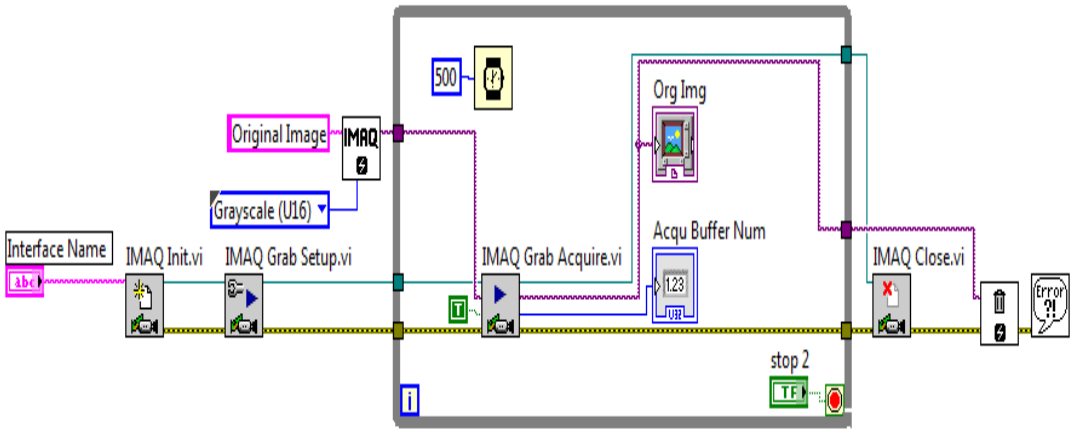


FIGURE 6. Vision development module.

MatLab Script Node

The MatLab script node is a module that is included with LabVIEW full development software for windows. It can be found on the formula VIs palette. The MatLab script node makes ActiveX calls to MatLab software from within LabVIEW. With this node, one can run the MatLab software programmatically from a LabVIEW program. You may enter the MatLab script in the node or import pre-existing MatLab code into the node.

Example: Interfacing of MatLab with LabVIEW. Below is the program to create random value generator. In the given example n=10 and the output b=0.533.

Shared Variables

LabVIEW provides access to a many varieties of technologies for creating distributed applications. Using the shared variable, you can share data between virtual instruments across the network or loops on a single diagram. You can create

three types of shared variables: time triggered, single-process, and network-published.

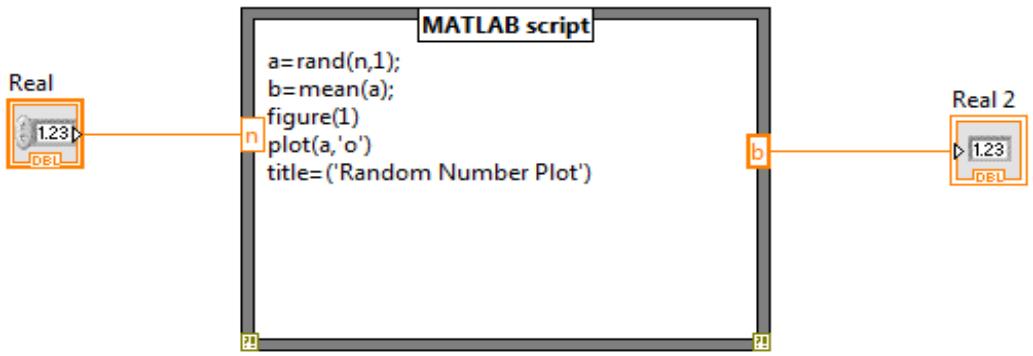


FIGURE 7. MatLab script design in LabVIEW.

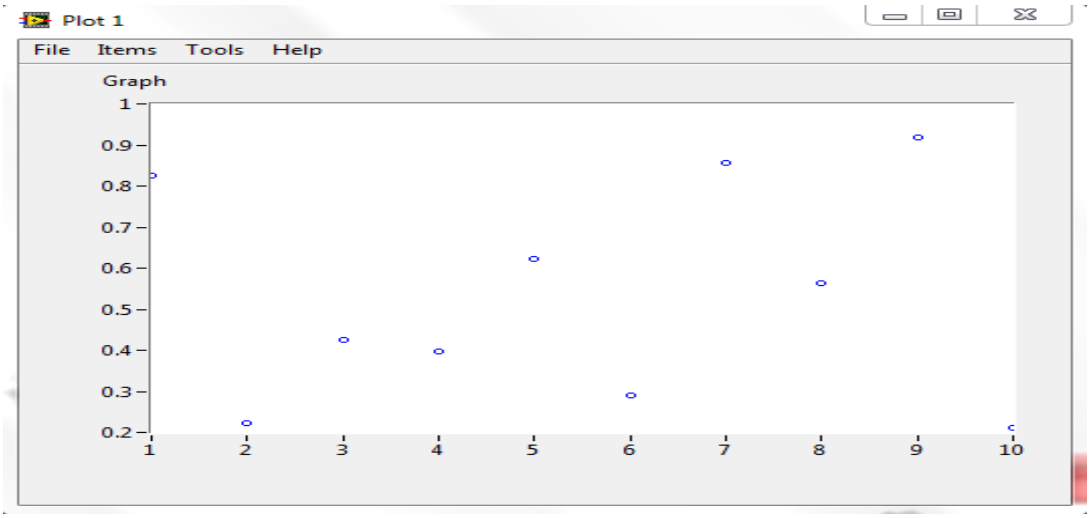


FIGURE 8. MatLab script results.

My project makes the use of network published shared variables. Using the network-published shared variable, you can use the same shared variable to

communicate data throughout the network. The shared variables over the network are used during network implementation.

In addition to making your data available over a network, a shared variable adds many features such as making your data available over the network. These features are not available with the single-process shared variable. In this functionality the implementation of the network-published shared variable is extensively more complex than the single process shared variable.

Deployment and hosting. Shared variable engine is used to host the shared variable values on the network. You must deploy network-published shared variables to it. When you write to a shared variable node, LabVIEW sends the new value to the shared variable engine that deployed the variable. The shared variable engine processing loop then publishes the value so that subscribers get the updated value. Shown below is network published shared variable process. To use client or server terminology, the shared variable engine is the server for a shared variable and all references are the clients, even if they write to or read from the variable. The shared variable engine client is part of the implementation of each shared variable node, and in this project the term subscriber and client are interchangeable.

MatLab

MatLab is a software package for visualization and high-performance numerical computation. It provides an interactive environment with many built-in extensible with its own high level programming language.

MatLab is an efficient program for matrix and vector data processing. It contains ready functions for image processing, visualization, matrix manipulation and allows a program to have modular structure. Due to these reasons MatLab has been chosen as

prototyping software. It provides a suitable environment for image and video processing.

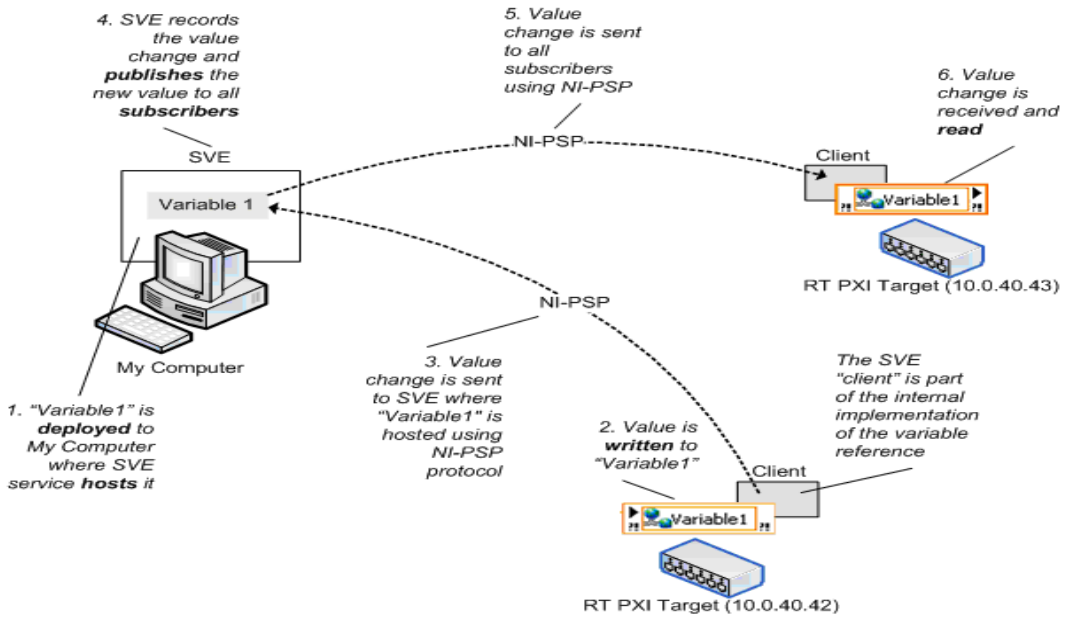


FIGURE 9. Shared variable engine and network shared variable process.

Although MatLab is slower than some languages (such as C and C++), it is built in functions and syntax makes it a faster and versatile programming environment for image processing. A programmer can change any previously written program to C or C++ using Active Gen (or another faster language) to make the program run faster.

Image Processing Using MatLab

MatLab is a general purpose programming language and it is used to process images. One generally writes script files, a function file to perform various operations. As these files make a formal record of the processing implemented and also ensures that the final results can be tested and by others should the need arise.

MatLab provides many functions for image processing and other tasks. Mostly these functions are written in the MatLab language and are publicly readable as plain text files and easily available to the user. Thus the implementation details of these functions are easily accessible. This makes MatLab different from other applications.

It should be noted that some MatLab functions cannot be viewed directly by the user. These are lower level functions that are computationally expensive and are provided as "built-in" functions running as native code. These functions are heavily used and can be relied on with considerable confidence.

One of the important advantages is that it allows one to ensure maximal numerical precision in the final result. While image files store data to 8-bit precision value. This corresponds to values ranging from 0 to 255. A pixel in a color image may be represented by three eight bit numbers, each representing the blue, red and green components as an integer value between 0 and 255. Typically this is ample precision (0 and 255) for representing normal images.

However as soon as one reads the image data into memory it starts the processing, it is very easy to generate values that lie outside of the range 0 to 255. For example in order to double the contrast of an image, one multiplies the intensity values by two. An image value of 250 will become 500 and numerical overflow will result. Handling of such type of image values will vary between image processing algorithms. Some of them may truncate the results to an integer with a range of 0 to 255; others may perform the mathematical operations in floating point arithmetic and then rescale the results to an integer in the range from 0 to 255.

It is here that numerical precision may be lost. Some image processing algorithms result in pixel values with large magnitudes. Typically these large values occur at points in the image where discontinued intensity occurs. When this image with

widely varying values is rescaled to integers in the range from 0 to 255 much of this range may be used just to represent the few pixels with the very large values. The bulk of the image data may have to be represented within a small range of integer values, say from 0 to 70. This can result in considerable loss of image information. If another process is applied to this image the problems can then accumulate.

As MatLab is a general programming language it is possible to have complete control of the precision with which one represents data. An image can be read and written from memory and the data cast into double precision floating point values. All image processing steps can be performed in double precision floating point arithmetic, and one does not need to rescale the results to integers in the range 0-255 at any stage. Only at the final point when the image is to be displayed or written to the file, it needs to be rescaled. The bulk of the image data is properly represented by eliminating external pixel value with the help of histogram.

MatLab provides strong mathematical and numerical support for the implementation of new algorithms. Hence MatLab is widely used by the image and video processing and computer vision community.

MatLab may not be as user friendly as an application like Photoshop; however, being a general purpose programming language it provides many advantages for forensic image processing.

It ensures the image processing steps used are completely documented, and hence can be reused. In general, the source code for all image processing functions are accessible for test. It ensures that the numerical precision is maintained all the way through the enhancement process. Image processing algorithms used in the MatLab are likely to be more advanced than those available from other image processing tools.

Image Representation Using MatLab

An image is stored as a matrix using standard MatLab conventions. There are five basic types of images supported by MatLab:

1. 8 - bit images
2. Intensity images
3. Binary images
4. RGB images
5. Indexed images

MatLab handles images as matrices. It means breaking each pixel of an image down into the elements of a matrix. MatLab can distinguish between grayscale and color images and therefore their resulting image matrices differ slightly.

A color is a composite of some rudimentary colors. MatLab therefore breaks each individual pixel of a color image down into blue, red, and green values. We get the result as, three matrices, one representing each color. All three matrices are stacked next to each other creating a three dimensional m by n by three matrixes.

For an image which has a height of five pixels and width of ten pixels the resulting in MatLab would be a five by ten by three matrixes for a true color image.

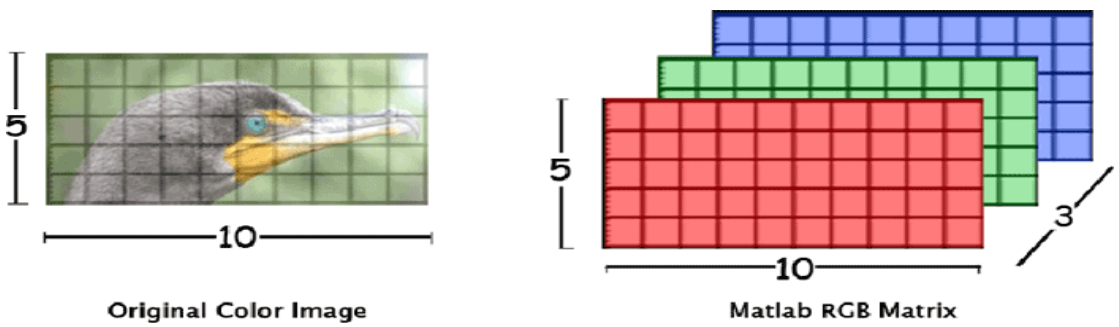


FIGURE 10. Color representation and RGB matrix.

A grayscale image is a combination of black and white colors. These colors, or as some may term as “shades,” are not composed of blue, red, and green colors. But instead they contain various increments of colors between white and black. Therefore to represent this single range, only single color channel is needed. Thus we only need a two dimensional matrix, m by n by one. MATLAB terms this type of matrix as an intensity matrix, this is due to the values of such a matrix they represent intensities of single color. For an image which as height of five pixels and width of ten pixels, the resulting matrix would be a five by ten matrix for grayscale image.

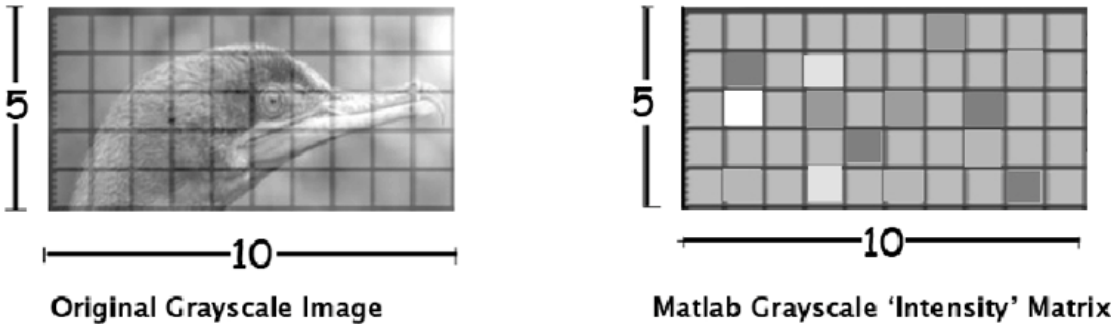


FIGURE 11. Grayscale representation and grayscale matrix.

CHAPTER 5

WORKING FLOW

Overview

The core algorithm is developed based on multiple studies that indicate how various parameters are used to determine the gender of a person. Although it is still unclear to say that these algorithms are 100% accurate and many recent researches are heading in that direction. The idea behind the algorithm lies in preprocessing the image and then reproducing the missing strokes in the image.

Explanation of Block Diagram

There are basically two ways to capture image into MatLab module, these can be given as:

Image from Webcam

Webcam. An external webcam is connected with the USB (Universal Serial Bus) port of the MyRIO FPGA kit. The webcam takes 640x480 size images and is kept in continuous image acquiring mode.

Image grab using vision acquisition module. Using the vision acquisition module from vision and motion package in LabVIEW the required camera source is selected and the image is imported using that. Here the camera holds cam0 name. The front panel of the VI shows the original image in real time and plays the video until the Image grab button is pressed present on the VI.

Block Diagram

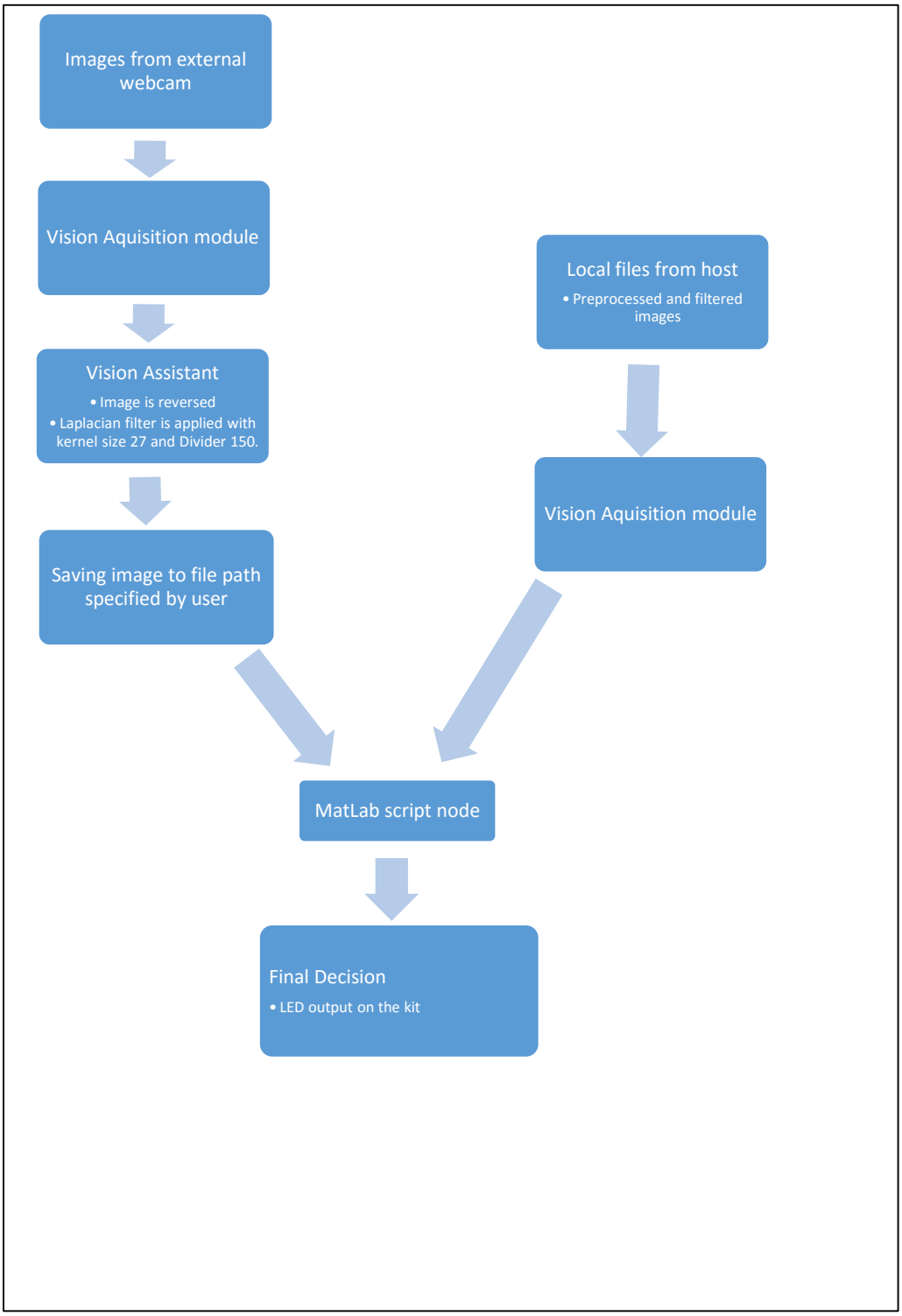


FIGURE 12. Overview of the working of process.

Preprocessing using vision assistant. The image then undergoes pre-processing using inbuilt functions in LabVIEW. First of all the image is reversed using the Look up table function. Various other factors such as brightness and contrast are adjusted. Secondly a Laplacian filter is applied to image using kernel value of 27 with divider value of 150.

Image from Locally Saved Files

Some signature samples were previously taken from the university library. These files are saved on the local computer. Hence the user suggests the file path and the image is grabbed from that file location.

MatLab Script Node

An inbuilt script node is used to call MatLab command window from the LabVIEW interface.

Final Decision

The final decision is dependent on the decision algorithm from the MatLab. The results are shown on the LEDs of the kit depicting whether the signature belongs to a male or female.

CHAPTER 6

ALGORITHM FOR DETECTION

Process in MatLab

MatLab Phase

The image undergoes number of processing to get the best results. Here is the detailed explanation that image undergoes during the detection process.

Processing and filtering for noise removal. The MatLab script node is provided with the file path where the shared variable stores the image after the pre-processing. This is fed into the "imread" function which reads the image. This is followed by the cropping function which crops the image for removing border noises. This does not affect the size of the image. This is then converted into binary image format. The binary image is then filtered using the "strel" function for "eye" opening to get the skeleton version of the image, followed by the "imerode" function. This removes most of the noise from the image and automatically crops only the signature part from the image.

Finding factor of intersections. Again using the cropped image, "imdilate" and "imerode" functions are used in combination to get a thin line image. A 3x3 kernel is applied to image and it is convoluted with itself for multiple times. This gives only the points where the intersection persists in the image.

Area of the image. The other factor responsible in detection is the area of the signature, which is obtained from the effective cropped image dimensions.

Image segmentation. The final decision factor requires considerable mathematical

processing of the image. The image is first segmented into six equal size segments. A “for” loop is used to iterate over each segment.

FIGURE 13. Process flow of MatLab phase.

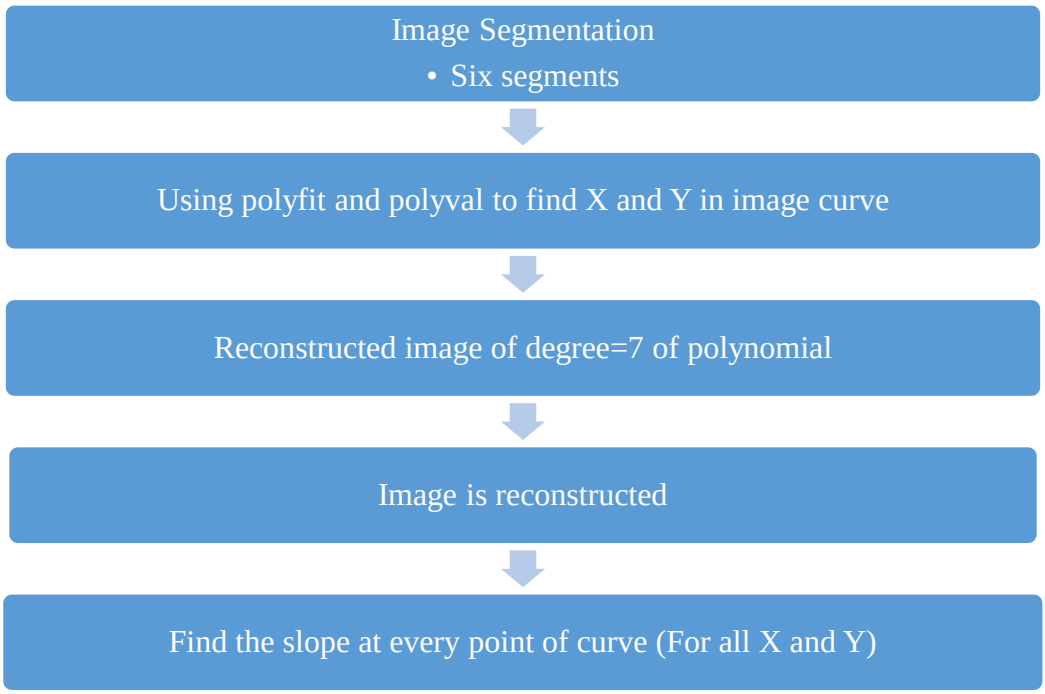


FIGURE 14. Image segmentation and reproduction flow.

Image reproduction. The x and y coordinates of each segments are obtained using the "polyfit" function and degree of n=7 for the polynomial function. This is then used for a "polyval" function to reproduce the image. The reconstructed image is used to find the slope at each and every point on the curve. This is final decision factor in detection algorithm.

CHAPTER 7

IMPLEMENTATION

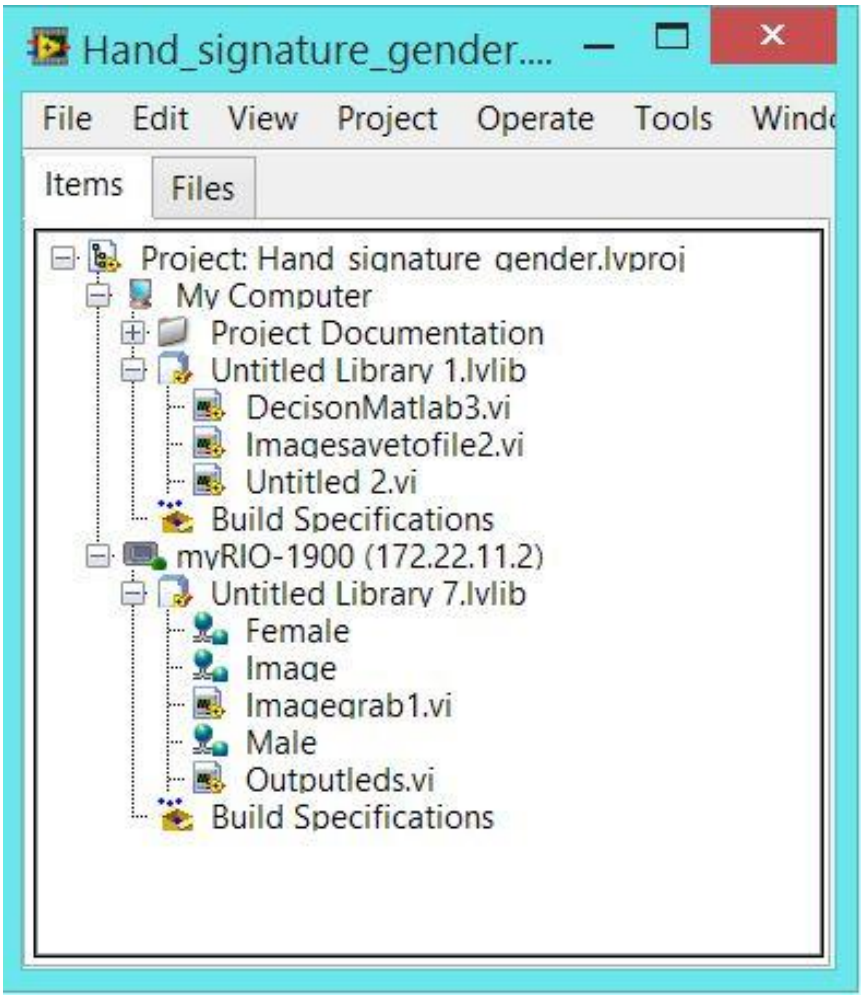


FIGURE 15. Project files window.

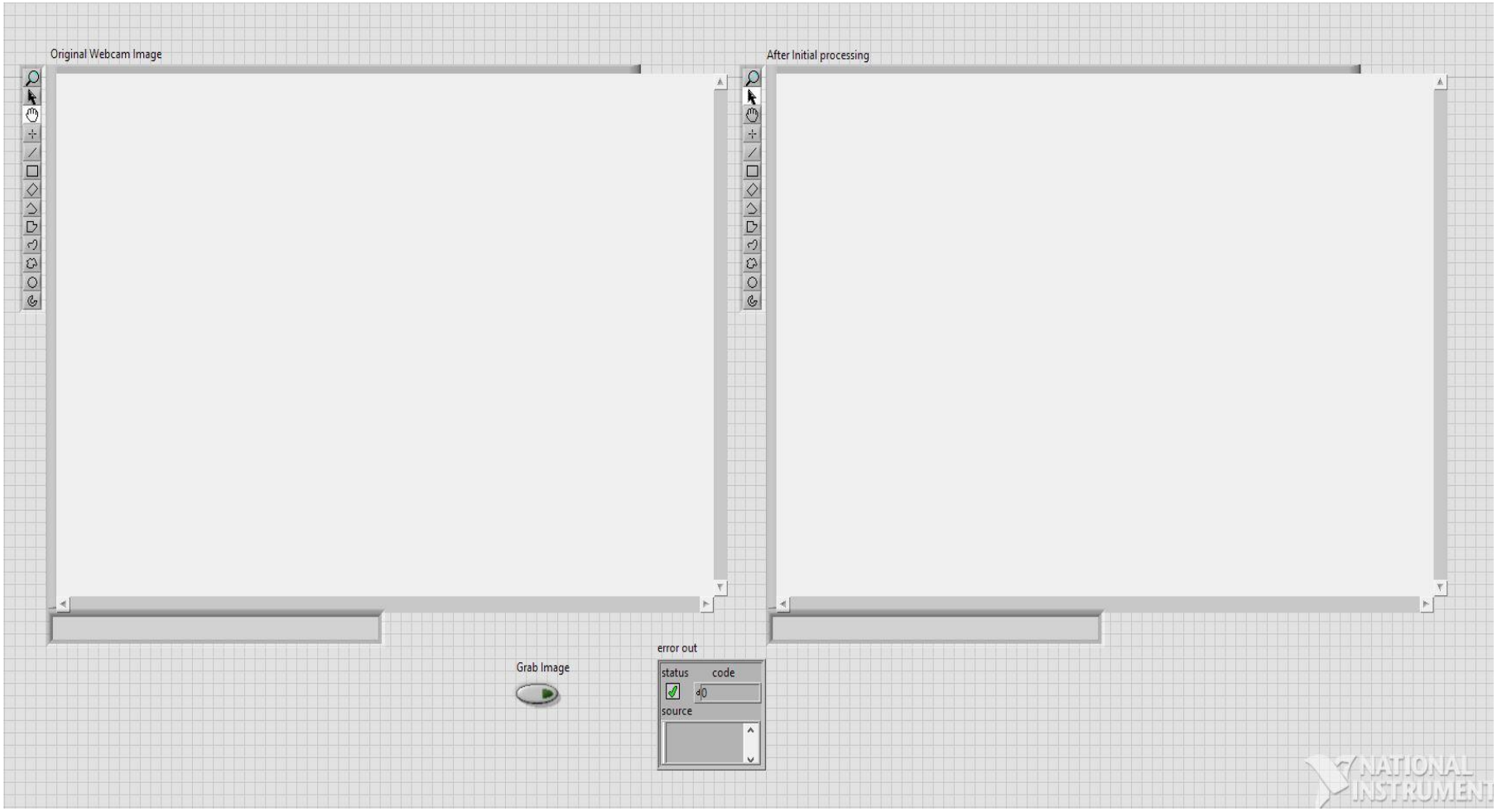


FIGURE 16. Front panel of image grab VI.

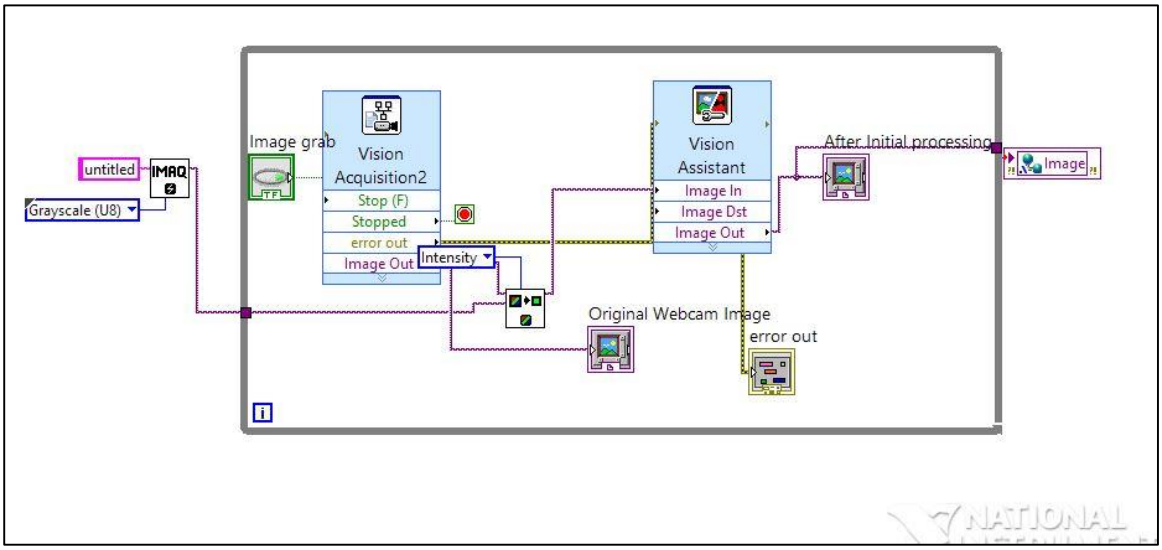


FIGURE 17. The graphical interconnections.

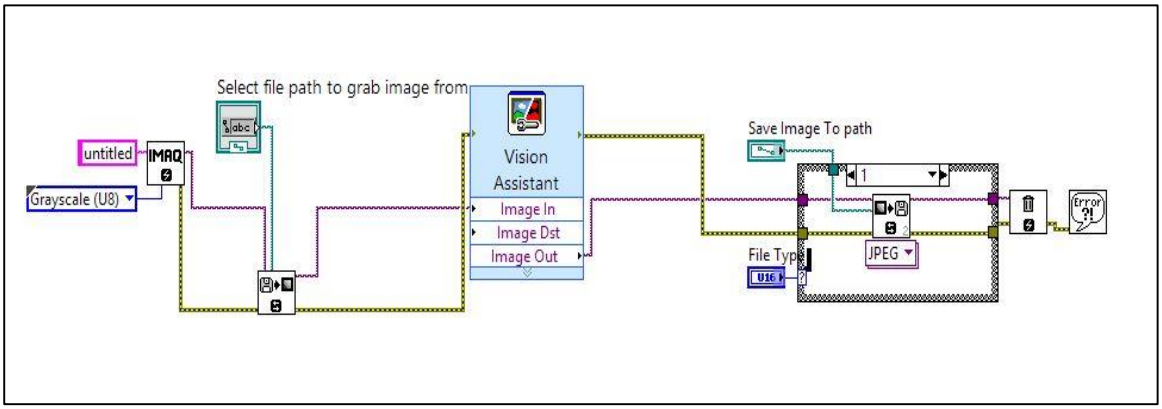


FIGURE 18. Grabbing from local files.

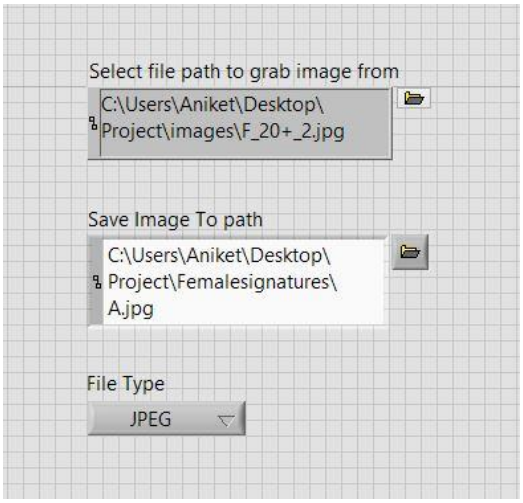


FIGURE 19. Saving to a path for local files.

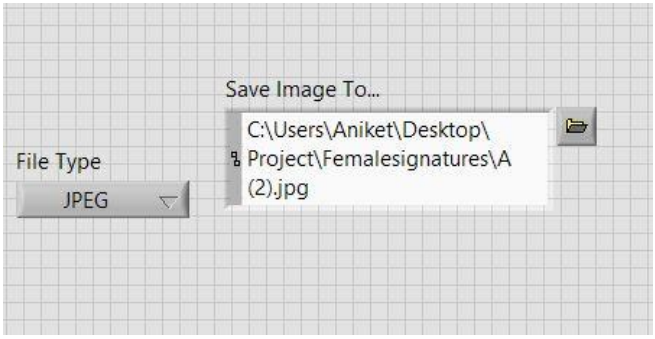


FIGURE 20. Saving to a path for webcam image.

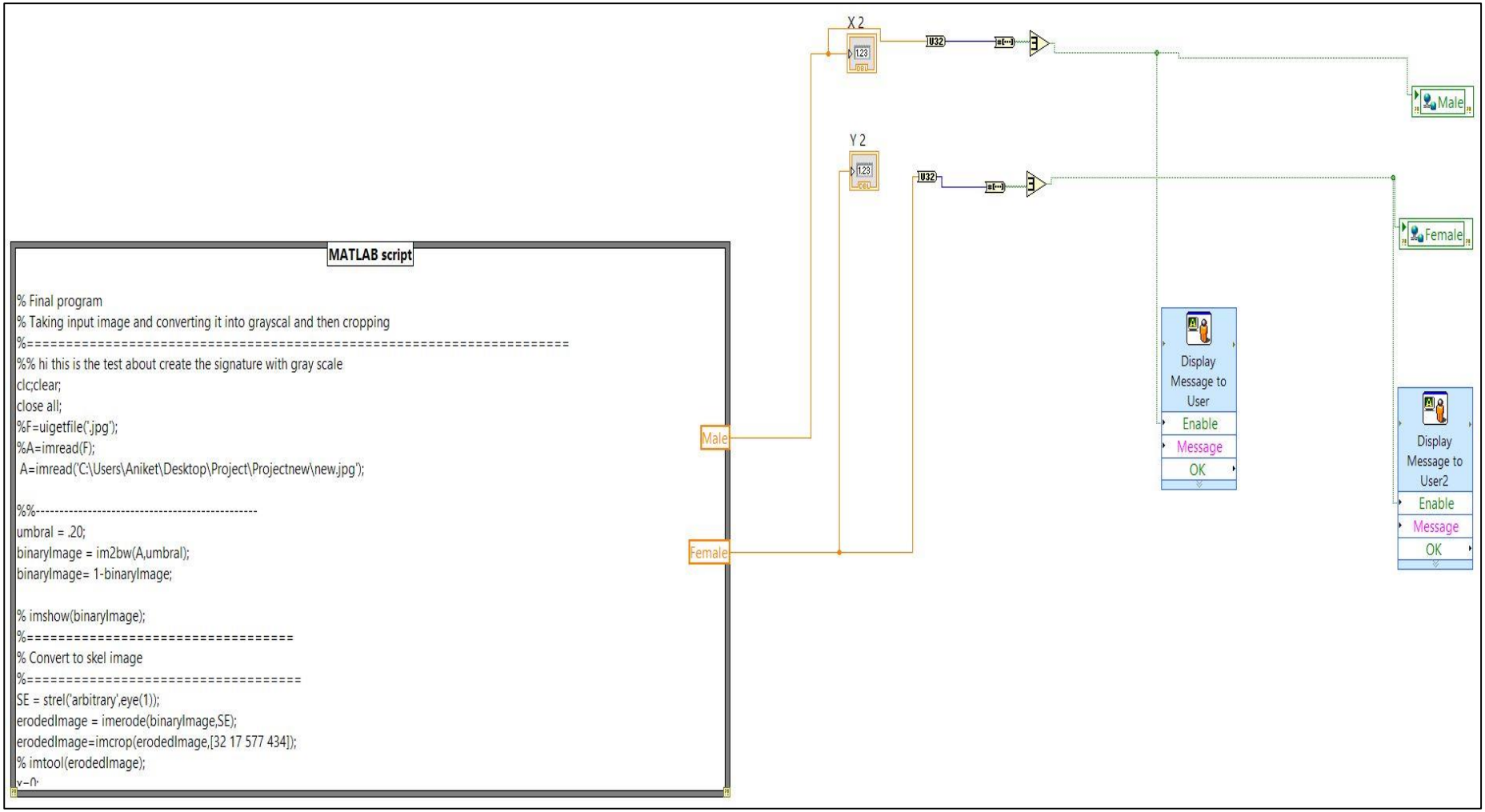


FIGURE 21. MatLab phase.

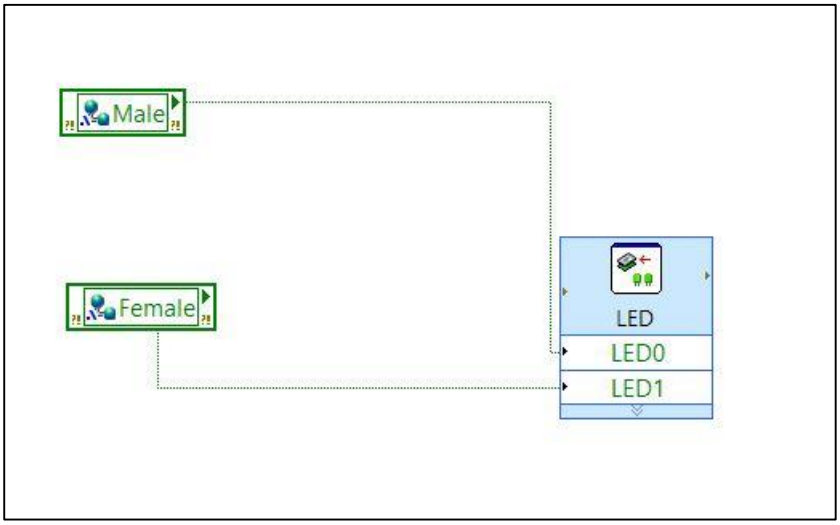


FIGURE 22. LED output on FPGA kit.

CHAPTER 8

RESULTS AND DISCUSSION

Observation

The results obtained from the detection are shown above. The first one depicts how male signatures are interpreted and the second one is for female signatures. The signatures are first converted into skeleton and then the final detection takes place. The system is about 70 % accurate and takes 2.9 seconds for the detection. Below is the table that shows how this system performs.

TABLE 1. Output Results and Consideration

1)	Accuracy Rate	70%
2)	Propagation Delay	2.9 Sec
3)	Operating Frequency	667 MHz
4)	FPGA Type	Xilinx ZYNQ Z-7010
5)	Output Voltage	4.75 V to 5.25 V
6)	Maximum current on each connector	100 mA

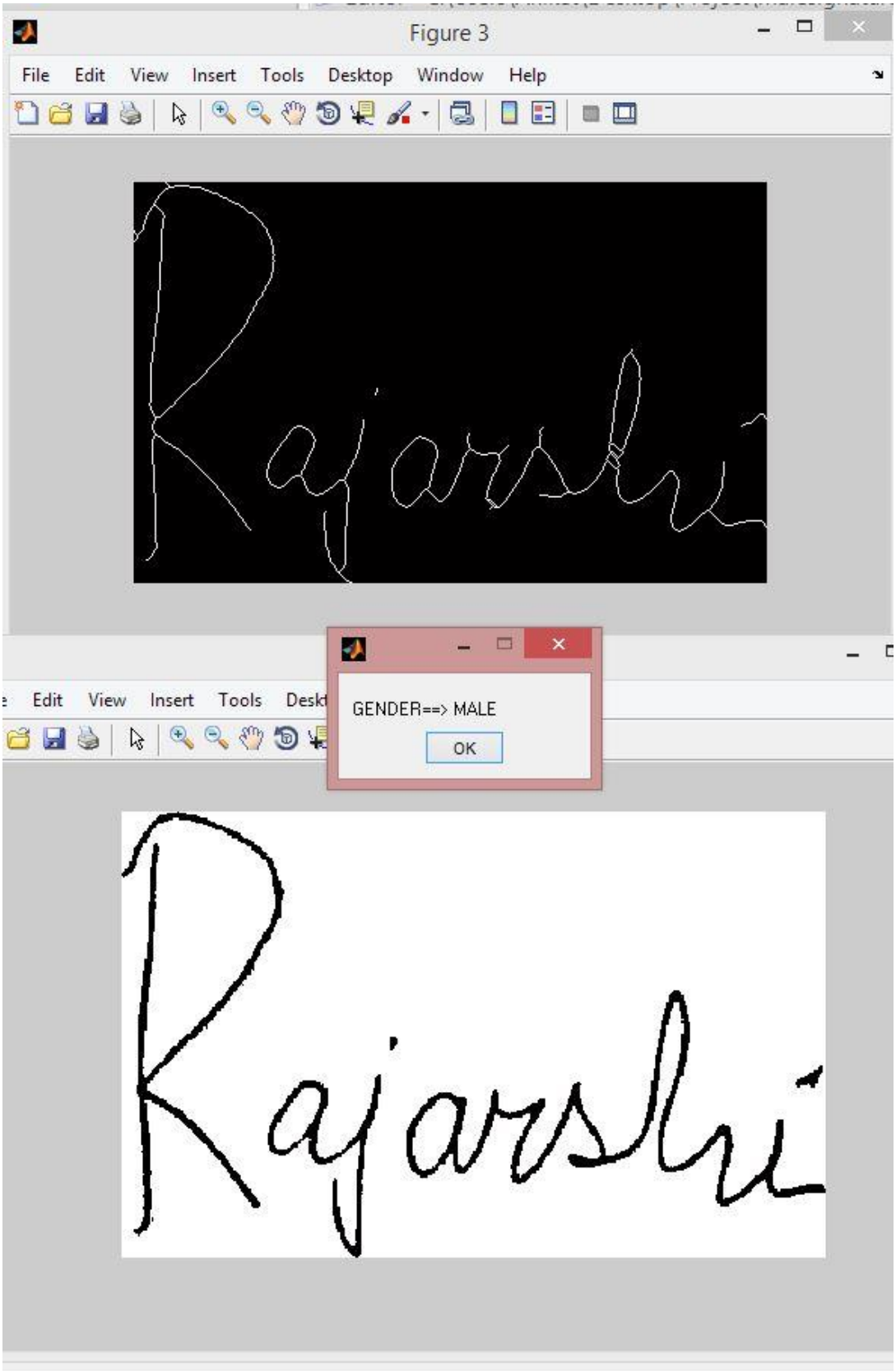


FIGURE 23. Male detection.

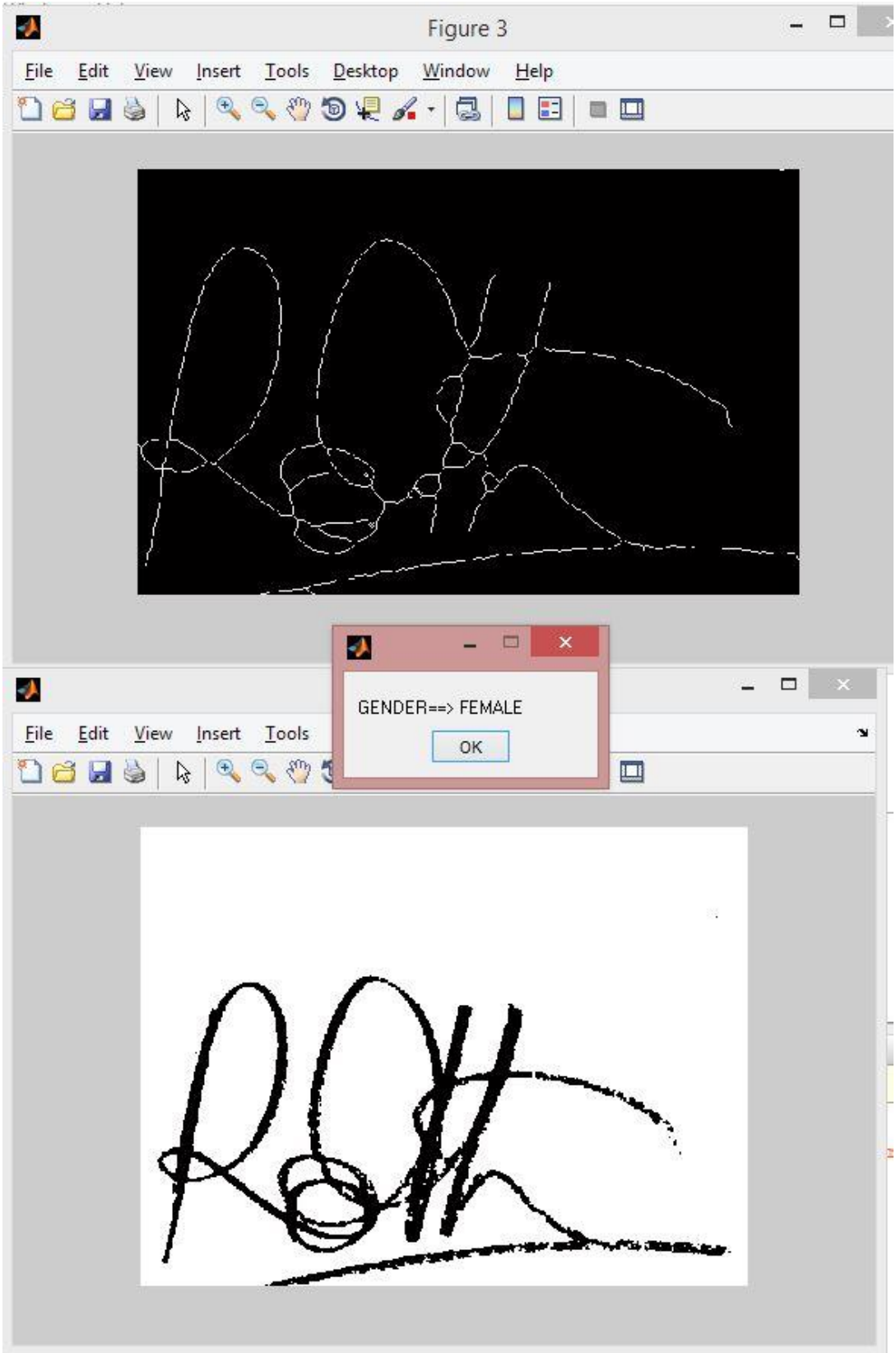


FIGURE 24. Female detection.

FIGURE 25. Output for male signature.

FIGURE 26. Output for female signature.

CHAPTER 9

ADVANTAGES AND LIMITATIONS

The system has essential advantages and limitations. The total dependence is on the size of image database, greater accuracy can be obtained with large database. Below are the listed advantages and limitations for this design. The demo requires taking image from an external webcam but in reality the signature is captured using pressure sensor pads placed at all stores.

Advantages

1. This system has major advantage in security protocols especially for the card authentication.

2. Moreover all the present embedded systems are hard programmed and cannot be reprogrammed with ease. The FPGA has the capability to be programmed remotely. Hence a FPGA that is connected over internet can be easily reprogrammed for any software updates.

3. This implementation can be extended to mobile devices. A unique signature can be added as a security feature for the payments and signing legal documents.

4. It is quite accurate and fast and takes only 2.9 seconds for the detection. But this is dependent on the system it is running on.

5. Power consumption is quite less and the works only on 5V.

6. The VI is error free and produces no error during or at the end of the execution.

Limitations

1. This system and algorithm has not any proven basis, as it is completely dependent on how the signatures strokes are done by the person.

2. Moreover it is also clear that signature is dependent on the age of the person. With increasing age the strokes get shaky and detection algorithm can fail in such scenarios.

3. Present system is not fully stand-alone as the execution still takes place on MatLab. The final system should be completely stand-alone and that would require programming completely in Linux environment.

CHAPTER 10

CONCLUSION

Cessation

The gender recognition using the hand signatures can change the way security protocols are implemented mainly by card companies. This technique of determining gender can drastically shorten the time to verify any signature from enormous database of all the users. Moreover the FPGA can replace all the embedded devices and can be updated over an internet remotely. This can also reduce computational load on a main server by distributing load over all remote devices.

Future Work

The future work will be dedicated towards improvement of the algorithm and database. This algorithm works accurately over larger database. Many thousands of signatures can be used for machine learning and can accurately produce results.

Main drawbacks of this project are:

1. Despite of multiple processing on LabVIEW and MatLab the image can still contain some noise, which can result in error during cropping.

2. Also it is dependent on perspective of the image, so if webcam is moved much closer or away from the picture it can result in error.

Although the drawback with the webcam is not a major concern as in real system the image is captured by a pressure sensor pad rather than a webcam. The webcam is implemented only to show demo how the algorithm will work.

APPENDICES

APPENDIX A

PROGRAM CODE FOR MATLAB

```
% Final program

%% Clearing and grabbing the image saved in the path
clc;clear;
close all;
A=imread('C:\Users\Aniket\Desktop\Project\MatLab Image Grab\A.jpg');
A=imcrop(A,[7 8 617 455]);

%% Using Umbral to enhance image and remove noise
umbral = .20;
binaryImage = im2bw(A,umbral); %% Umbral = level (0.5 is equal white and black)
binaryImage= 1-binaryImage;

%% Convert to skeleton image for better interpretation

SE = strel('arbitrary',eye(1));
erodedImage = imerode(binaryImage,SE);
erodedImage=imcrop(erodedImage,[32 17 577 434]);
figure,imshow(erodedImage);

%% Cropping the image automatically from size of signature

x=0;
pp=0;
y=0;
qq=0;
m=0;
ss=0;
n=0;
rr=0;
[rows cols]=size(erodedImage);

v=0;
```

```
for j=1:cols
for i=1:rows

   if erodedImage(i,j)==0
x=j;
v=v+1;
if(v==1)
   pp=x;
end
   end

end

end

v=0;
i=0;
j=0;
for j=1:cols
   for i=1:rows
     if erodedImage(i,j)==0
    qq=j;
     end
   end
end

v=0;
i=0;
j=0;
for i=1:rows
   for j=1:cols
       if erodedImage(i,j)==0
           m=i;
```

```
            v=v+1;
            if(v==1)
               rr=m;
               end
         end
      end
end

v=0;
i=0;
j=0;
for i=1:rows
   for j=1:cols
      if erodedImage(i,j)==0
         ss=i;
      end
   end
end

xmin=rr;
ymin=pp;
height=ss-rr;
width=qq-pp;
new_image=imcrop(erodedImage,[ymin xmin width height]);
saved_image=new_image;
gray_image=new_image;
figure,imshow(gray_image);


%% Taking image and converting it into binary image and skelImage for number of
intersections

umbral=0.50;
```

```
binaryImage = im2bw(new_image,umbral);
binaryImage= 1-binaryImage;

SE = strel('square',5);
dilatedImage = imdilate(binaryImage,SE);
SE=strel('arbitrary',eye(4));
erodeimg=imerode(dilatedImage,SE);

skelImage=bwmorph(erodeimg,'thin',Inf);
figure,imshow(skelImage)
Z=0;O=0;

%% 1) FACTOR --- Calculating number of intersections


   isi = double(skelImage) ./ double(max(skelImage(:)));
   kernel = [1 1 1; 1 1 1; 1 1 1];
   conv_img = conv2(isi,kernel,'same');
   conv_img2 = conv_img .* isi;

   [Y,X] = find(conv_img2 > 3);

   numberPoints = length(X);

   if numberPoints>152 && numberPoints<71
   Z=Z+0.10;
   end
   if numberPoints>51 && numberPoints<175
   O=O+0.10;
   end

%% 2) FACTOR--- Calculating size of sign
[X1 Y1] = size(new_image);
```

```
Area= X1*Y1;
if Area<202000 && Area>150000 || (Area >0 && Area <60000)
   Z=Z+0.40;
else
   O=O+0.40;
end

%% Dividing image in eight sections for reconstruction
[numrows numcols]=size(skelImage);
for j=1:floor(numcols/8)
   for i=1:numrows

      R1(i,j)=skelImage(i,j);
   end
end

for j=floor(numcols/8):floor(numcols/4)
   for i=1:numrows

      R2(i,j)=skelImage(i,j);
   end
end

for j=floor(numcols/4):floor(3*numcols/8)
   for i=1:numrows

      R3(i,j)=skelImage(i,j);
   end
end

for j=floor(3*numcols/8):floor(numcols/2)
   for i=1:numrows
```

```
        R4(i,j)=skelImage(i,j);
    end
end

for j=floor(numcols/2):floor(5*numcols/8)
    for i=1:numrows

        R5(i,j)=skelImage(i,j);
    end
end

for j=floor(5*numcols/8):floor(6*numcols/8)
    for i=1:numrows

        R6(i,j)=skelImage(i,j);
    end
end

for j=floor(6*numcols/8):floor(7*numcols/8)
    for i=1:numrows

        R7(i,j)=skelImage(i,j);
    end
end

for j=floor(7*numcols/8):numcols
    for i=1:numrows

        R8(i,j)=skelImage(i,j);
    end
end

R=[R1 R2 R3 R4 R5 R6 R6 R7 R8];
```

```
%% Check for the pixels and get their locations
x=1;
y=1;[a b]=size(R6);
curv=0;
slope=0;
m=1;

for i=1:a
   for j=1:b
      temp_img=R6;
         if temp_img(i,j)==1
         x(m)=[i];
         y(m)=[j];
         m=m+1;
      end
    end
end

figure,imshow(R6);
n=7;
p=polyfit(x,y,n);
f=polyval(p,x);
figure,plot(x,f,'-',x,y,'*');
y1=0;

reg_image=zeros(size(R6));

%% 3) Calculate the slope at all such points and get amount of curvature in
%     the strokes.


for l=1:m-1
```

```
   if l<m-1
   slope(l)=y(l+1)-y(l)/x(l+1)-x(l);
   end
end

%% Factor of curvature

for w=1:l-1
     if (slope(w) >55) || (slope(w) <10)
          curv=curv+1;
end
end

if curv>270
   Z=Z+0.10;
else
   O=O+0.10;
end
Male=0;
Female=0;

%% Decision
if Z>= 0.20
   msgbox('GENDER==> FEMALE');
Female=5;
else
   msgbox('GENDER==> MALE');
Male=3;
end
```

APPENDIX B

MYRIO SPECIFICATIONS

The following specifications are typical for the 0 to 40 °C operating temperature range unless

Otherwise noted.

- Processor

Processor type .....Xilinx Z-7010

Processor speed.....667 MHz

Processor cores 2

- Memory

Nonvolatile memory .....256 MB

DDR3 memory.....512 MB

DDR3 clock frequency .....533 MHz

DDR3 data bus width.....16 bits

For information about the lifespan of the nonvolatile memory and about best practices for using

nonvolatile memory, go to ni.com/info and enter the Info Code SSDBP.

- FPGA

FPGA type .....Xilinx Z-7010

- Wireless Characteristics

Radio mode .....IEEE 802.11 b,g,n

Frequency band.....ISM 2.4 GHz

Channel width .....20 MHz

- USB Ports

USB host port..... USB 2.0 Hi-Speed

USB device port..... USB 2.0 Hi-Speed

- Analog Input

Aggregate sample rate..... 500 kS/s

Resolution..... 12 bits

Overvoltage protection ..... ±16 V

MXP connectors

Configuration..... Four singl/e-ended channels per connector

Input impedance ..... >500 kΩ acquiring at 500 kS/s

1 MΩ powered on and idle

4.7 kΩ powered off

Recommended source impedance ..... 3 kΩ or less

Nominal range ..... 0 V to +5 V

Absolute accuracy..... ±50 mV

Bandwidth..... >300 kHz

MSP connector

Configuration.....Two differential channels

Input impedance ..... Up to 100 nA leakage powered on;

4.7 kΩ powered off

Nominal range ..... ±10 V

Working voltage

(signal + common mode)..... ±10 V of AGND

Absolute accuracy..... ±200 mV

Bandwidth..... 20 kHz minimum, >50 kHz typical

Audio input

Configuration..... One stereo input consisting of two AC-coupled,

single-ended channels

Input impedance ..... 10 kΩ at DC

Nominal range ..... ±2.5 V

Bandwidth..... 2 Hz to >20 kHz

- Analog Output

Aggregate maximum update rates

All AO channels on MXP connectors.....345 kS/s

All AO channels on MSP connector

and audio output channels.....345 kS/s

Resolution .....12 bits

Overload protection .....±16 V

Startup voltage .....0 V after FPGA initialization

MXP connectors

Configuration .....Two single-ended channels per connector

Range .....0 V to +5 V

Absolute accuracy.....50 mV

Current drive .....3 mA

Slew rate .....0.3 V/µs

MSP connector

Configuration .....Two single-ended channels

Range .....±10 V

Absolute accuracy.....±200 mV

Current drive .....2 mA

Slew rate .....2 V/µs

Audio output

Configuration .....One stereo output consisting of two AC-coupled, single-ended channels

Output impedance .....100 Ω in series with 22 µF

Bandwidth.....70 Hz to >50 kHz into 32 Ω load; 2 Hz to >50 kHz into high-impedance load

- Digital I/O

Number of lines

MXP connectors .....2 ports of 16 DIO lines (one port per connector); one UART.RX and one UART.TX line per connector

MSP connector.....1 port of 8 DIO lines

Direction control .....Each DIO line individually programmable as input or output

Logic level .....5 V compatible LVTTL input; 3.3 V LVTTL output

Input logic levels

Input low voltage, VIL ..... 0 V min; 0.8 V max

Input high voltage, VIH ..... 2.0 V min; 5.25 V max

Output logic levels

Output high voltage, VOH

sourcing 4 mA ..... 2.4 V min; 3.465 V max

Output low voltage, VOL

sinking 4 mA ..... 0 V min; 0.4 V max

Minimum pulse width..... 20 ns

Maximum frequencies for secondary digital functions

SPI .....4 MHz

PWM..... 100 kHz

Quadrature encoder input ..... 100 kHz

I2C..... 400 kHz

UART lines

Maximum baud rate..... 230,400 bps

Data bits..... 5, 6, 7, 8

Parity..... Odd, Even, Mark, Space

Flow control.....XON/XOFF

- Accelerometer

Number of axes..... 3

Resolution..... 12 bits

Sample rate ..... 800 S/s

Noise..... 3.9 mgrms typical at 25 °C

- Power Output

+5 V power output

Output voltage ..... 4.75 V to 5.25 V

Maximum current on each connector ..... 100 mA

+3.3 V power output

Output voltage ..... 3.0 V to 3.6 V

Maximum current on each connector ..... 150 mA

+15 power output

Output voltage.....+15 V to +16 V

Maximum current .....32 mA (16 mA during startup)

-15 V power output

Output voltage.....-15 V to -16 V

Maximum current .....32 mA (16 mA during startup)

Maximum combined power from +15 V
and -15 V power output .....500 mW

- Power Requirements

NI MyRIO-1900 requires a power supply connected to the power connector.

Caution: You must use either the power supply provided in the shipping kit, or another UL Listed ITE power supply marked *LPS*, with the NI MyRIO-1900.

Power supply voltage range.....6-16 VDC
Maximum power consumption .....14 W
Typical idle power consumption.....2.6 W

- Environmental

To meet these specifications, you must operate the NI MyRIO-1900 with the window facing
away from the mounting surface and ensure that there is at least 1 in. of clearance in front of the
window during use.

Ambient temperature near device
(IEC 60068-2-1, IEC 600682-2).....0 to 40 °C
Storage temperature
(IEC 60068-2-1, IEC 600682-2).....-20 to 70 °C
Operating humidity (IEC 60068-2-56) .....10 to 90% RH, noncondensing
Storage humidity (IEC 60068-2-56) .....10 to 90% RH, noncondensing
Maximum altitude.....2,000 m
Pollution Degree (IEC 60664) 2
Indoor use only.

- Physical Characteristics

Weight .....193 g (6.8 oz)

BIBLIOGRAPHY

BIBLIOGRAPHY

Chapran, J. 2006. "Biometric Writer Identification: Feature Analysis and Classification." *International Journal of Pattern Recognition & Artificial Intelligence* 20: 483–503.

Faundez-Zanuy, Marcos. 2007. "On-line Signature Recognition Based on VQ-DTW." *Pattern Recognition* 40 (3): 981–992.

Houmani, Nesmaa, A. Mayoue, S. Garcia-Salicetti, B. Dorizzi, M.I. Khalil, M. Mostafa, H. Abbas, Z.T. Kardkovàcs, D. Muramatsu, B. Yanikoglu, A. Kholmatov, M. Martinez-Diaz, J. Fierrez, J. Ortega-Garcia, J. Roure Alcobé, J. Fabregas, M. Faundez-Zanuy, J. M. Pascual-Gaspar, V. Cardeñoso-Payo, and C. Vivaracho-Pascual. March 2012. "BioSecure Signature Evaluation Campaign (BSEC'2009): Evaluating online signature algorithms depending on the quality of signatures." *Pattern Recognition* 45 (3): 993–1003.

MathWorks. 2015. "MATLAB: The Language of Technical Computing." Accessed Dec 21, 2014. www.mathworks.com/ products/matlab.

National Instruments. 2015. "Test, Measurement, and Embedded Systems." Accessed January 19. 2015. www.ni.com.

Ortega-Garcia, Javier, J. Fierrez, D. Simon, J. Gonzalez, M. Faúndez-Zanuy, V. Espinosa, A. Satue, I. Hernaez, J.-J. Igarza, C. Vivaracho, D. Escudero, and Q.-I. Moro. 2003. "MCYT Baseline Corpus: A Multimodal Biometric Database." *IEE Proceedings - Vision, Image and Signal Processing* 150: 395–401.

Said, H. E. S., T. N. Tan, and K. D. Baker. 2000. "Personal Identification Based on Handwriting." *Pattern Recognition* 33: 149–160.

Schomaker, L. 2007. "Advances in Writer Identification and Verification." In *Proceedings Ninth International Conference on Document Analysis and Recognition (*ICDAR*)*: 1268–1273. New York: IEEE Computer Society.

Schlapbach, A., M. Liwicki, and H. Bunke. 2008. "A Writer Identification System for Online Whiteboard Data." *Pattern Recognition* 41 (7): 2381–2397.

Sesa-Nogueras, Enric, and Marcos Faundez-Zanuy. 2012. "Biometric Recognition Using Online Uppercase Handwritten Text". *Pattern Recognition* 45 (1): 128–144.

Yeung, D.H., Y. Xiong, S. George, R. Kashi, T. Matsumoto, and G. Rigoll. 2004. "SVC2004: First International Signature Verification Competition." In *Lecture Notes in Computer Science (*LNCS-3072*)*, 16–22. Hong Kong: Springer.