EMOTION DETECTION USING GALVANIC SKIN RESPONSE METER

Dissertation submitted in partial fulfillment of the requirement for the degree of

BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMMUNICATION ENGINEERING

By
Vanesh Trikha 121071
Ayush Kanwar 121097
Abhinab Thakur 121104

UNDER THE GUIDANCE OF
Dr. Pradeep Kumar

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT
June 2016

TABLE OF CONTENTS

DECLARATION BY THE SCHOLAR

We hereby declare that the work reported in the B-Tech Thesis entitled **"EMOTION DETECTION USING GALVANIC SKIN RESPONSE METER"** submitted at **Jaypee University of Information Technology, Waknaghat, India,** is an authentic record of our work carried out under the supervision of **Dr. Pradeep Kumar.** We have not submitted this work elsewhere for any other degree or diploma.

Vanesh Trikha (121071)

Ayush Kanwar (121097)

Abhinab Thakur (121104)

Department of Electronics & Communication

Jaypee University of Information Technology, Waknaghat , India

Date :

SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the B-Tech. thesis entitled **"EMOTION DETECTION USING GALVANIC SKIN RESPONSE METER "** submitted by **Vanesh Trikha (121071), Ayush Kanwar (121097) and Abhinab Thakur(121104)** at **Jaypee University of Information Technology, Waknaghat , India,** is a bonafied record of their original work

carried out under my supervision. This work has not been submitted elsewhere for any other degree or diploma.

Dr. Pradeep Kumar
Assistant Professor (Senior Grade)

Date:

ACKNOWLEDGEMENT

It is said that gratitude is a virtue. This part is dedicated to special thanks that we would like to deliver to the people who help make possible the fulfillment of this project.

First of all, we would like to thank Dr. Pradeep Kumar for introducing us to this intriguing research field. Thank you for the motivation and enlightening insights. Without your support and guidance we would not have completed our project. We are especially grateful that you treat us like a friend, respect our decisions and plans, and encourage us with every progress, no matter how humble it is, along the way.

We would also like to thank Mr. Dhirendra K. Singh and Mr. Manoj Pandey for helping us in the labs. Thank you for allowing us to work in your labs and sparing your precious time for us.

LIST OF ACRONYMS

1. EDA : Electrodermal Activity
2. GSR : Galvanic Skin Response
3. GUI : Graphic User Interface

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Emotion recognition is an important part of the affective computing research field. Physiological signals could reflect the true feelings of people more objectively, because they couldn't be controlled subjectively.

The Galvanic Skin Response (GSR) signal, has been applied to many areas of research and people's real life. It changes with the change of the body's sympathetic nervous system, depending on the body's secreted response of sweat glands. Secretion of sweat is affected by the body's emotional arousal, the temperature, and activities of finger, etc.

The aim of our project is to implement a basic circuit that can generate data through Electrodermal Conductance (Galvanic Skin Response) for detecting five emotions: Angry, Sad, Happy (Excited), Fear, Neutral.

Our method is to extract features from physiological signal's morphological changes according to the fitting parameters, and it provides a new idea for the emotion recognition based on physiological signals.

.

CHAPTER 1

INTRODUCTION

1.1 **BASIC EDA SYSTEM**

Galvanic Skin Response (GSR) or **Electrodermal Activity (EDA)** is the measurement of the electrical conductance of the skin, which varies depending on the amount of sweat-induced moisture on the skin. GSR is considered to be a function of sweat gland activity. Sweat is a weak electrolyte and a good conductor, so GSR is measured by applying a low voltage electric current to the skin. As the body comes under stress, sweat production increases, and the sweat ducts fill which affect the skin conductance.

The traditional theory of EDA holds that skin resistance varies with the state of sweat glands in the skin. Sweating is controlled by the sympathetic nervous system, and skin conductance is an indication of psychological or physiological arousal. If the sympathetic branch of the autonomic nervous system is highly aroused, then sweat gland activity also increases, which in turn increases skin conductance. In this way, skin conductance can be a measure of emotional and sympathetic responses. More recent research and additional phenomena (resistance, potential, impedance, and admittance, sometimes responsive and sometimes apparently spontaneous) suggest this is not a complete answer, and research continues into the source and significance of EDA.

An EDA recording can be drawn like this :-

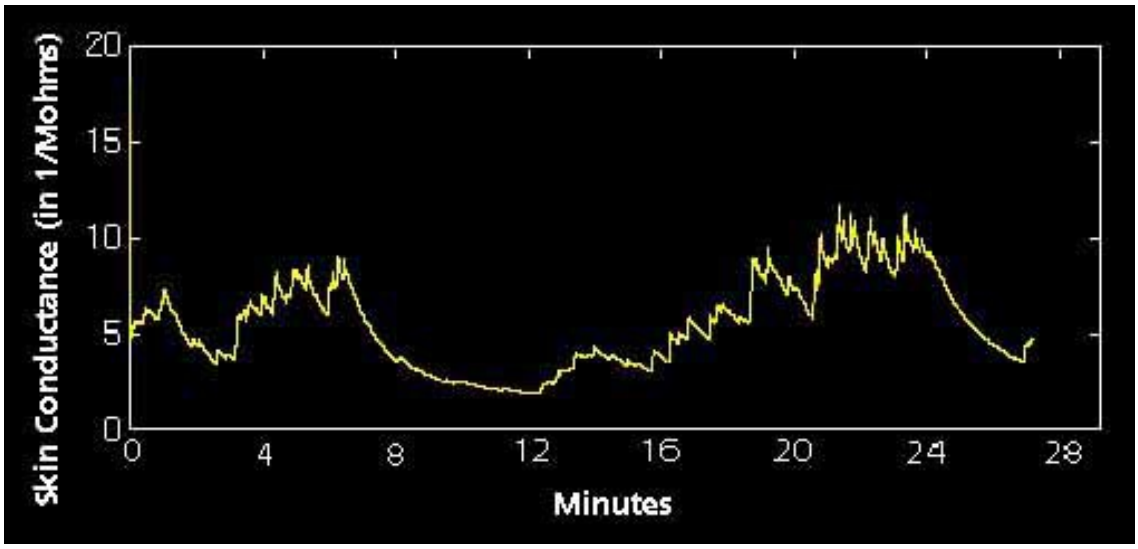

Figure 1.1: An EDA Waveform

CHAPTER 2

REVIEW/BACKGROUND MATERIAL

2.1 SWEAT GLANDS

There are three types of sweat glands, Eccrine, Apocrine and Apoeccrine.

Eccrine glands are approximately 3 million of the nearly 4 million sweat glands. Their main function is to control body temperature. Areas of the body that have the highest concentration of Eccrine glands are the soles of the feet, palms of the hands, and the forehead. These are the sweat glands responsible for sweaty palms. When they're stimulated, they release a clear odorless fluid that evaporates into the air, thereby cooling you off when your body temperature gets too high.

Apocrine Sweat glands primarily reside in the armpits and the genital regions. They produce a thicker, oily, odorless, fluid that gets released into the hair follicles. Bacteria that reside on the skin break down this fluid and the result is the bad odor if not showered regularly. In mammals, this oily fluid is thought by some to act as a pheromone to attract a mate, or act as a warning signal to other mammals.

Apoeccrine glands also primarily reside in the armpits and genital regions. They develop during puberty from what are known as Eccrine precursors, and can produce sweat up to seven times faster than Eccrine glands. The role of these types of glands is controversial, but it's thought to be involved in the excessive sweating of people with a condition known as Axillary Hyperhidrosis (excessive armpit sweating).
The sweat glands are activated by a type of nerve fiber called a sympathetic fiber. These are the nerves that react to the sympathetic nervous system. When this fight or flight system becomes active, sweat occurs.

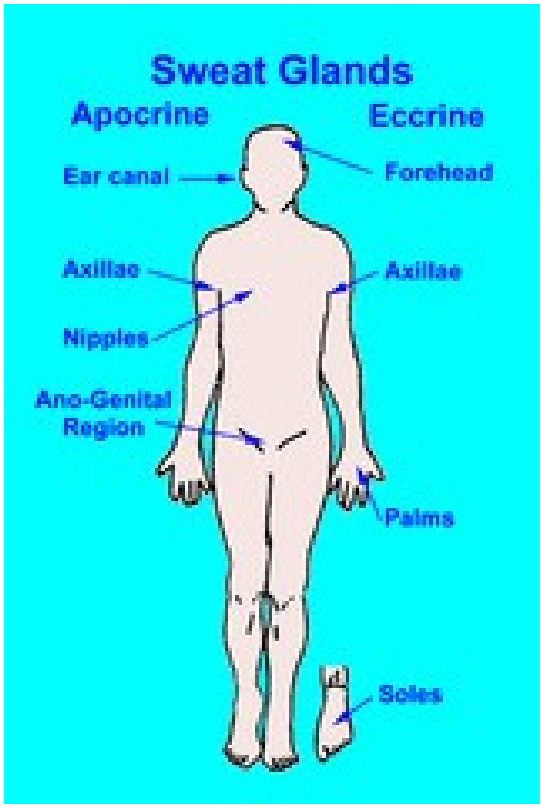

Figure 2.1: Sweat Glands Distribution

2.2 Skin Resistance

A rough value for the *internal* resistance of the human body is 300-1,000 Ohms. Naturally, the resistance also depends on the path that electricity takes through the body - if the electricity goes in the left hand and out the right foot, then the resistance will be much higher than if it goes in and out of adjacent fingers.

Within the body, the tissues with the greatest resistance are bone and fat - nerves and muscle have the least resistance. That said, the majority of the body's resistance is in the skin - the dead, dry cells of the epidermis (the skin's outer layer) are very poor conductors. Depending on the person, the resistance of *dry skin* is usually between 1,000-100,000 Ohms. The skin's resistance is much lower if it is wet or burnt/blistered. This means that when a person is electrocuted in real life, the body's resistance drops as the skin is burned. To determine a person's total resistance, just add together the resistance of each part of the body - remember that the electricity must pass through the skin twice (on the way in and on the way out), so the total resistance is:

Rtotal = Rskin(in) + Rinternal + Rskin(out)

Another interesting point to consider is that in addition to acting like a resistor, the epidermis acts like a capacitor if placed in contact with a piece of metal (the underlying tissue is like one plate of a capacitor and the metal surface is like the other plate - the dry epidermis is the less-conductive material or "dielectric" in between) . In cases of electrocution by a DC voltage source, this capacitive property has little importance. But if the electrocution is by an AC source, the epidermis's natural resistance is "shorted out", allowing the current to bypass that part of the body's resistance and making the body's total resistance much lower.

CHAPTER 3

HARDWARE USED

3.1 Arduino ATMega2560

The Arduino Mega is a microcontroller board. It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

Figure 3.1: Arduino Mega Board

Table 3.1: Arduino Specifications

Microcontroller	ATmega1280
Operating Voltage	5V
Input Voltage (recommended)	7-12V

Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	128 KB of which 4 KB used by boot loader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

3.2 Programming

The Arduino Mega can be programmed with the Arduino software.

The ATmega1280 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer.

It communicates using the original STK500 protocol.

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header.

1. 470 KOhms register
2. Aluminium foil
3. Velcro
4. Jumper cables, connecting wires

3.3 Softwares Used

3.3.1 MATLAB R2008a

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment. A proprietary programming language developed by MathWorks. MATLAB allows matrix manipulation, plotting of functions and data, implementation of algorithms, creation of user interfaces and interacting with programs written in other languages including C, C++, Java, Fortran and Python.

GUIs (also known as graphical user interfaces or UIs) provide point-and-click control of software applications, eliminating the need to learn a language or type commands in order to run the application.

MATLAB apps are self-contained MATLAB programs with GUI front ends that automate a task or calculation. The GUI typically contains controls such as menus, toolbars, buttons, and sliders. Many MATLAB products, such as Curve Fitting Toolbox, Signal Processing Toolbox, and Control System Toolbox, include apps with custom user interfaces.

GUIDE (graphical user interface design environment) provides tools for designing user interfaces for custom apps. Using the GUIDE Layout Editor, one can graphically design your UI. GUIDE then automatically generates the MATLAB code for constructing the UI, which you can modify to program the behavior of the app.

3.3.2 Arduino Software (IDE)

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

3.3.3 Block Diagram

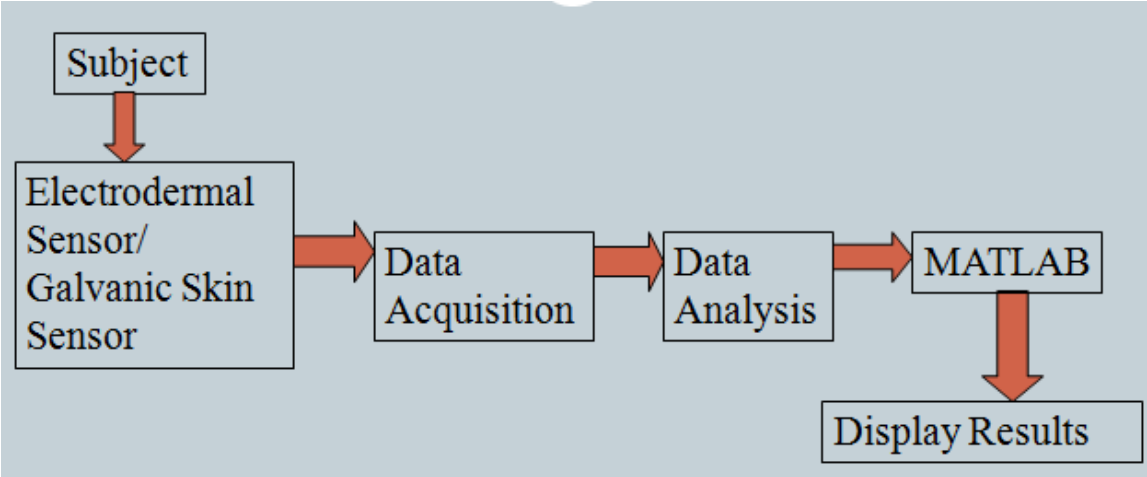

Figure 3.2: Block diagram

3.3.4 Circuit Diagram

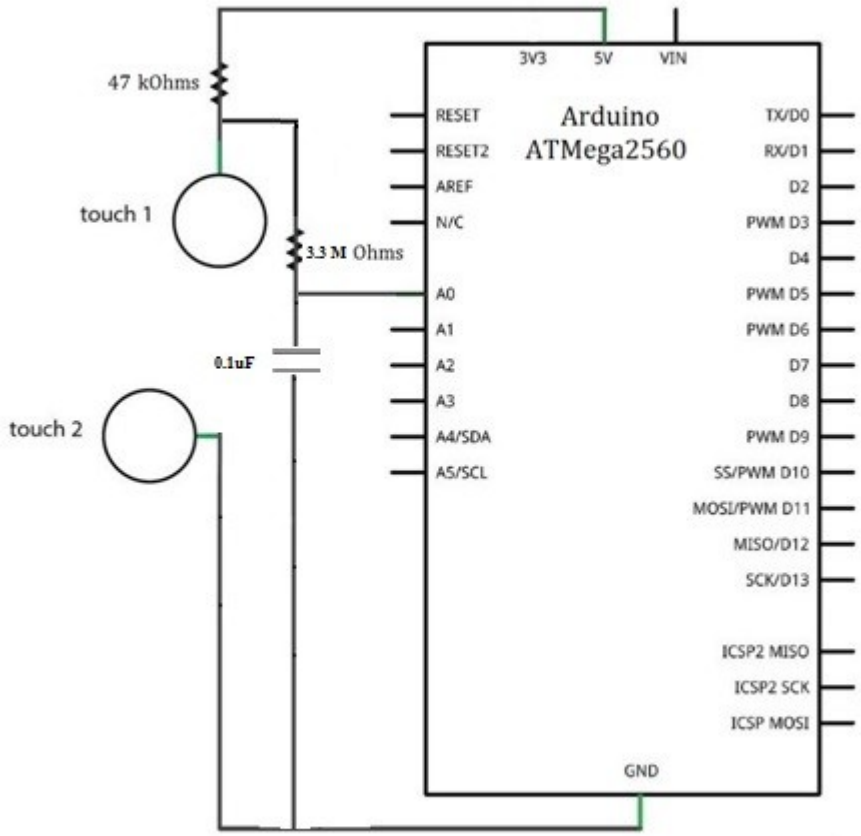

Figure 3.3: Circuit Diagram

CHAPTER 4

WORK DESCRIPTION

4.1 Setting of Experiment

To recognize emotion based on GSR signal, we would collect the GSR signal in a particular emotional state first. In our study, we used short segments of movies and short films to elicit the target emotions and record the emotional physiological signals to set up the database of our experiment.

4.2 Procedure

1. Select one person from your group to be the subject. Ask the subject to go to the sink, wash his or her hands with soap and water, and dry them thoroughly. Washing the hands insures that surface oils or other substances, which might lower skin conduction, are removed. Do not use alcohol to clean the fingers, alcohol dehydrates the skin.

2. Connect the GSR electrodes to the circuit.

3. The subject should sit with his or her back to the computer monitor.

4. Attach each GSR electrode to the volar surface of the distal finger segment of two adjacent fingers. Attach the electrodes with the Velcro straps so that the straps are snug, but not overly tight.

5. The subject should rest his or her hand with the GSR electrodes comfortably. The GSR electrodes should be free from any extraneous pressure and the electrode cable should be hanging freely. Instruct the subject not to move the hand during the recording process; movement will introduce artifacts into the recording.

6. Two measurements are performed in this exercise:

   • Tonic Skin Conductance Level: In this measurement, the subject sits quietly for one minute as his or her tonic skin conductance level (SCL) is recorded. Any movement may cause an artifact in the recording.

   • Habituation: In this measurement, the subject's tonic skin conductance level (SCL) is recorded as he or she sits quietly and without movement. While recording the subject's SCL, ask the subject the question: Is your name, X? (where X is the subject's real name). The subject should respond: Yes. After the subject's SCL changes and then returns to the baseline, ask the subject the same question. Continue to ask this same question until the subject shows no response on three consecutive trials. Lack of a change in the subject's SCL while answering the question is considered habituation.

7. Finally, the recording of data can be started now.

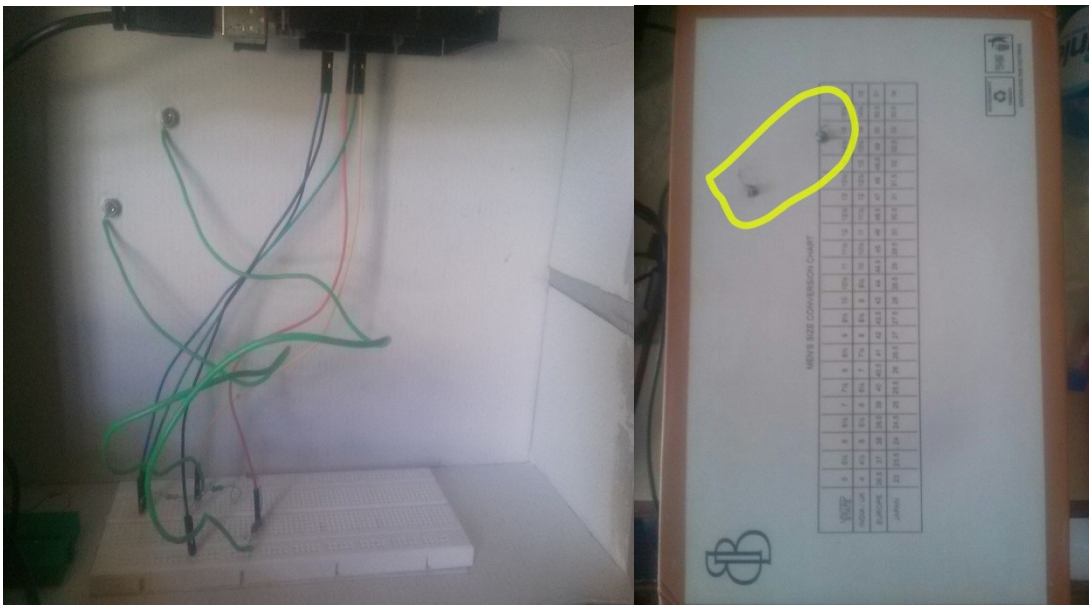

Figure 4.1: Circuit Setup

CHAPTER 5

CODE

5.1 Code for Arduino Software(IDE) (to be uploaded in Arduino Board)

```
#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
#define INTERNAL INTERNAL1V1
#endif
void setup()
```

```
{
   Serial.begin(115200);
}
void loop()
{
 static int  s   = -1;
 static int  pin = 13;
  int  val =  0;
 int  agv =  0;
 int  dgv =  0;
 if (Serial.available() >0)
 {
   val = Serial.read();
   switch (s) {
    case -1:
    if (val>47 && val<90) {
     s=10*(val-48);
    }
    if ((s>40 && s<90) || (s>90 && s!=340 && s!=400)) {
     s=-1;
    }

    case 0:
    if (val>98 && val<167) {
     pin=val-97;
     s=1;      }
    else {
     s=-1;
    }
    break;
    case 1:
    if (val>47 && val<50) {
     if (val==48) {
      pinMode(pin,INPUT);
     }
     else {
      pinMode(pin,OUTPUT);
     }
    }
    s=-1;
 break;
    case 10:
    if (val>98 && val<167) {
     pin=val-97;
```

```
      dgv=digitalRead(pin);
      Serial.println(dgv);
    }
    s=-1;
    break;


case 20:
    if (val>98 && val<167) {
      pin=val-97;
      s=21;
    }
    else {
      s=-1;
    }
    break;
    case 21:
      if (val>47 && val<50) {
      dgv=val-48;
digitalWrite(pin,dgv);
    }
    s=-1;
    break;
       case 30:
    if (val>96 && val<113) {
      pin=val-97;
      agv=analogRead(pin);
Serial.println(agv);
    }
    s=-1;       break;
    case 40:
    if (val>98 && val<167) {
      pin=val-97;
      s=41;
    }

else {
      s=-1;     }
    break;
   case 41:
        analogWrite(pin,val);
 s=-1;
    break;
    case 90:
```

```
     if (val==57) {
       Serial.println(0);
     }
     s=-1;
     break;
    case 340:
     #if defined(__AVR__) || defined(__PIC32MX__)
     switch (val) {
          case 48:
       analogReference(DEFAULT);
       break;
         case 49:
       analogReference(INTERNAL);
       break;
           case 50:
       analogReference(EXTERNAL);
       break;
         default:
       break;
     }

#endif
     s=-1;
     break;
       case 400:
     Serial.println(val);
     s=-1;
     break;
      default:
        s=-1
   }
  }
}
```

5.2 Code for GUI on MATLAB R2008a

```
function varargout = IR(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @IR_OpeningFcn, ...
                   'gui_OutputFcn',  @IR_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
```

```
                'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function IR_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
 delete(instrfind({'Port'},{'com28'}))
clear a;
global a;
a= arduino('com28');
 function varargout = IR_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function read_button_Callback(hObject, eventdata, handles)
global a k j l;
x=0;
y=0;
 R=470000;
h=figure();
handles.xSamples = 5*handles.xTime;
for k=0:0.2:handles.xTime
    b=a.analogRead(0);
    c=b*(5/1024);
    B=((5-c)/((c)*R))*1000000;
    y=[y,c];
    z=[z,f];
    x=[x,k];
    plot(x,y); grid on;
     xlabel('Time (s)');
    ylabel('Conductance ( uS )');
    saveas(h,'response','fig')
    pause(0.1);
end
function edit_text_samples_Callback(hObject, eventdata, handles)
handles.data1=get(hObject,'String');
handles.xTime=str2double(handles.data1);
guidata(hObject,handles);
function edit_text_samples_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

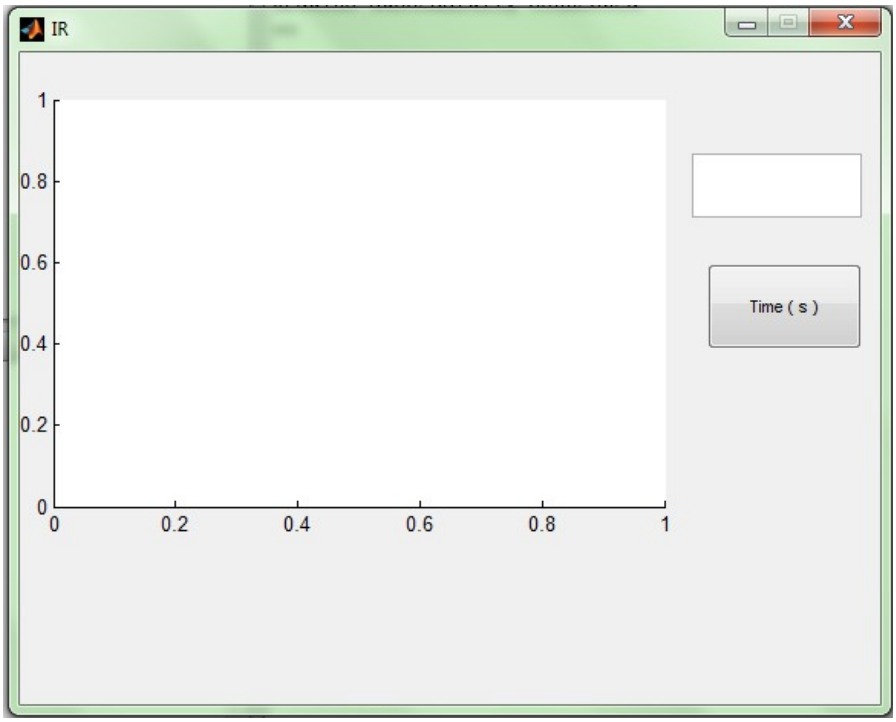

Figure 5.1: GUI in MATLAB R2008a

5.3 Code for Data Analysis

```
open('response.fig');
i=gcf;
axesObjs=get(i,'Children');
dataObjs=get(axesObjs,'Children');
xdata=get(dataObjs,'XData');
ydata=get(dataObjs,'YData');
prompt1 = 'Limit 1 : ';
r1= input(prompt);
prompt2 = 'Limit 2 :';
r2= input(prompt2);
prompt3 = 'Limit 3 :';
r3= input(prompt3);
prompt4 = 'Limit 4 :';
r4= input(prompt4);
R1 = (ydata(r1)*5 ) + 2 ;
R2 = (ydata(r2)*5 ) + 2 ;
R3 = (ydata(r3)*5 ) + 2 ;
```

```
R4 = (ydata(r4)*5 ) + 2 ;
Y1=ydata(R1:R2);
Y2=ydata(R3:R4);
L1=length(Y1);
L2=length(Y2);
Y11=sum(Y1)/L1;
Y22=sum(Y2)/L2;
COR=0;
for j=1:1:L1;
COR=COR + ((Y1(j)-Y11)*(Y2(j)-Y22));
end
E1=0;
E2=0;
for l=1:1:L1
E1= E1 + (Y1(l)-Y11)^2;
E2= E2 + (Y2(l)-Y22)^2;
end
NC=COR/(sqrt(E1*E2));
```

CHAPTER 6

CURVE FITTING

Curve Fitting is the process of constructing a curve, or mathematical function that has the best fit to a series of data points, possibly subject to constraints. Curve fitting can involve either interpolation, where an exact fit to the data is required,

or smoothening, in which a "smooth" function is constructed that approximately fits the data.

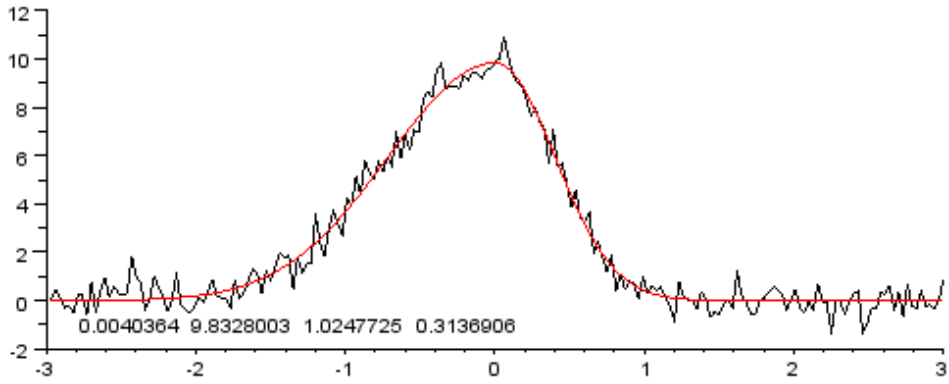

Figure 6.1: Curve Fitting

6.1 Interpolation

In the mathematical field of numerical analysis, interpolation is a method of constructing new data points within the range of a discrete set of known data points.
In engineering and science, one often has a number of data points, obtained by sampling or experimentation, which represent the values of a function for a limited number of values of the independent variable. It is often required to interpolate (i.e. estimate) the value of that function for an intermediate value of the independent variable. This may be achieved by curve fitting or regression analysis.

6.2 Smoothening

To smooth a data set is to create an approximating function that attempts to capture important patterns in the data, while leaving out noise or other fine-scale structures/rapid phenomena. In smoothing, the data points of a signal are modified so individual points (presumably because of noise) are reduced, and points that are lower than the adjacent points are increased leading to a smoother signal.
Smoothing may be used in two important ways that can aid in data analysis –

1. By being able to extract more information from the data as long as the assumption of smoothing is reasonable.

2. By being able to provide analyses that are both flexible and robust.

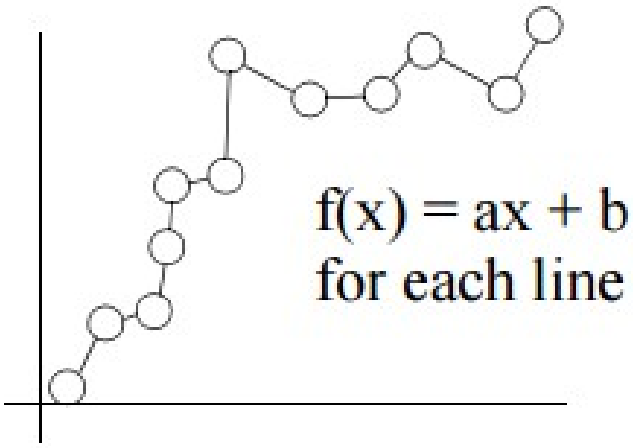

Figure 6.2: Interpolation

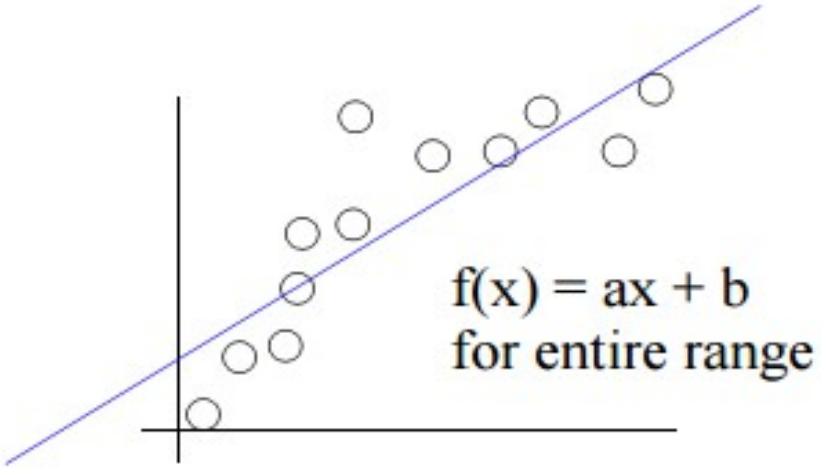

Figure 6.3: Smoothening

6.3 Curve Fitting in MATLAB

MATLAB® lets us import and visualize our data, and perform basic fitting techniques such as polynomial and spline interpolation. One can perform data fitting interactively using the MATLAB Basic Fitting tool, or programmatically using MATLAB functions for fitting.

MATLAB add-on products extend data fitting capabilities to:

- Fit curves and surfaces to data using the functions in Curve Fitting Toolbox™. Several linear, nonlinear, parametric, and nonparametric models are included. You can also define your own custom models.
- Fit N-dimensional data using the linear and nonlinear regression capabilities in Statistics and Machine Learning Toolbox™. You can also use machine learning algorithms for data-driven fitting.
- Perform constrained data fitting where parameters need to satisfy linear or nonlinear constraints with Optimization Toolbox™.

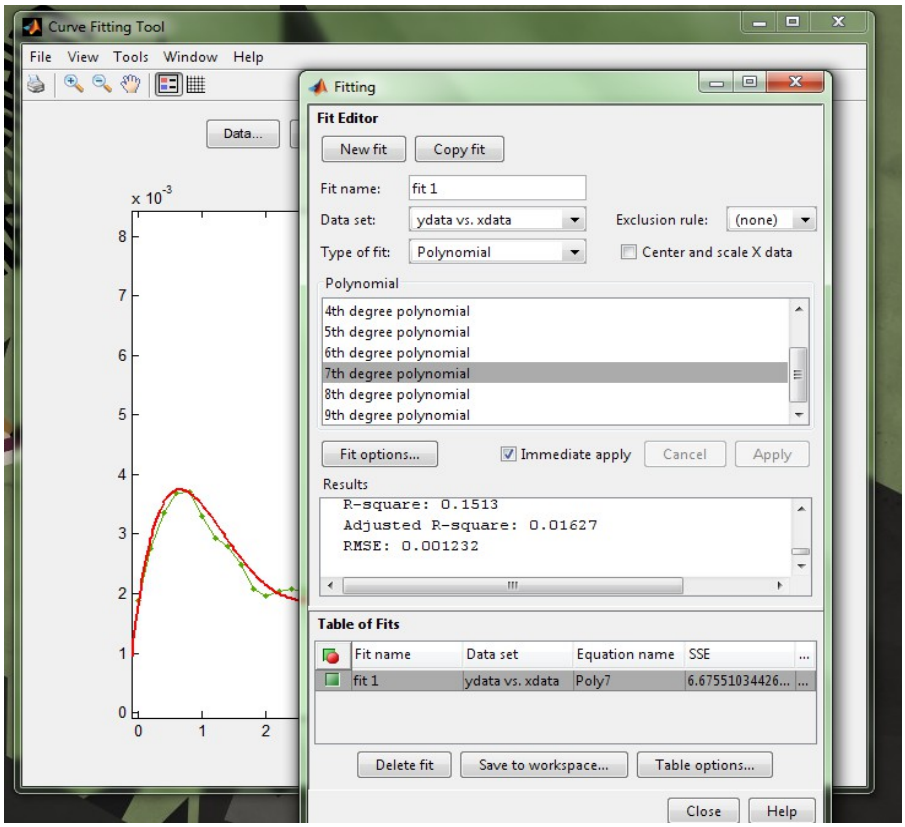

Figure 6.4: Curve Fitting Toolbox™

6.4 **Signal Smoothing in MATLAB**

Smoothing is how we discover important patterns in our data while leaving out things that are unimportant (i.e. noise). We use filtering to perform this smoothing. The goal of smoothing is to produce slow changes in value so that it's easier to see trends in our data.

Sometimes when you examine input data you may wish to smooth the data in order to see a trend in the signal.

CHAPTER 7

TYPES OF SIGNAL SMOOTHING IN MATLAB

7.1 Moving Average Filter

A moving average filter smoothes data by replacing each data point with the average of the neighbouring data points defined within the span. This process is equivalent to lowpass filtering with the response of the smoothing given by the difference equation :

$$y_s(i)=12N+1(y(i+N)+y(i+N-1)+...+y(i-N))$$

where $y_s(i)$ is the smoothed value for the ith data point, N is the number of neighbouring data points on either side of $y_s(i)$, and $2N+1$ is the span.

The moving average smoothing method used by Curve Fitting Toolbox™ follows these rules:

- The span must be odd.
- The data point to be smoothed must be at the centre of the span.
- The span is adjusted for data points that cannot accommodate the specified number of neighbours on either side.
- The end points are not smoothed because a span cannot be defined.

7.2 Filter Delay

The filtered output is delayed by a few samples. This is due to the fact that our moving average filter has a delay.
Any symmetric filter of length N will have a delay of (N-1)/2 samples. We can account for this delay manually.

7.3 Weighted Moving Average Filter

Other kinds of moving average filters do not weight each sample equally.
Another common filter follows the binomial expansion of (1/2,1/2)^n This type of filter approximates a normal curve for large values of n. It is useful for filtering out high frequency noise for small n. To find the coefficients for the binomial filter, convolve [1/2 1/2] with itself and then iteratively convolve the output with [1/2 1/2] a prescribed number of times.
Another filter somewhat similar to the Gaussian expansion filter is the exponential moving average filter. This type of weighted moving average filter is easy to construct and does not require a large window size.
You adjust an exponentially weighted moving average filter by an alpha parameter between zero and one. A higher value of alpha will have less smoothing.

7.4 Savitzki-Golay Filter

Savitzky-Golay filtering can be thought of as a generalized moving average. You derive the filter coefficients by performing an unweighted linear least-squares fit using a polynomial of a given degree. For this reason, a Savitzky-Golay filter is also called a digital smoothing polynomial filter or a least-squares smoothing filter. Note that a higher degree polynomial makes it possible to achieve a high level of smoothing without attenuation of data features.
The Savitzky-Golay filtering method is often used with frequency data or with spectroscopic (peak) data. For frequency data, the method is effective at preserving the high-frequency components of the signal. For spectroscopic data, the method is effective at preserving higher moments of the peak such as the line width. By comparison, the moving average filter tends to filter out a significant portion of the signal's high-frequency content, and it can only preserve the lower moments of a peak such as the centroid. However, Savitzky-Golay filtering can be less successful than a moving average filter at rejecting noise.

The Savitzky-Golay smoothing method used by Curve Fitting Toolbox™ software follows these rules:

- The span must be odd.
- The polynomial degree must be less than the span.
- The data points are not required to have uniform spacing.

Normally, Savitzky-Golay filtering requires uniform spacing of the predictor data. However, the Curve Fitting Toolbox™ algorithm supports nonuniform spacing. Therefore, you are not required to perform an additional filtering step to create data with uniform spacing.

7.5 Median Filter

Moving average, weighted moving average, and Savitzky-Golay filters smooth all of the data they filter. This, however, may not always be what is wanted. For example, what if our data is taken from a clock signal and has sharp edges that we do not wish to smooth? The filters discussed so far do not work so well:

The moving average and Savitzky-Golay filters respectively under-correct and over-correct near the edges of the clock signal.

A simple way to preserve the edges, but still smooth the levels is to use a median filter.

7.6 Local Regression Smoothing

- Lowess and Loess
- The Local Regression Method
- Robust Local Regression

7.6.1 Lowess and Loess

The names "lowess" and "loess" are derived from the term "locally weighted scatter plot smooth," as both methods use locally weighted linear regression to smooth data.

The smoothing process is considered local because, like the moving average method, each smoothed value is determined by neighboring data points defined within the span. The process is weighted because a regression weight function is defined for the data points contained within the span. In addition to the regression weight function, you can use a robust weight function, which makes the process resistant to outliers. Finally, the methods are differentiated by the model used in the regression: lowess uses a linear polynomial, while loess uses a quadratic polynomial.

The local regression smoothing methods used by Curve Fitting Toolbox™ software follow these rules:

- The span can be even or odd.

- You can specify the span as a percentage of the total number of data points in the data set. For example, a span of 0.1 uses 10% of the data points.

7.6.2 The Local Regression Method

The local regression smoothing process follows these steps for each data point:

1. Compute the *regression weights* for each data point in the span. The weights are given by the tricube function shown below.

$$w_i = \left(1 - \left|\frac{x - x_i}{d(x)}\right|^3\right)^3$$

x is the predictor value associated with the response value to be smoothed, x_i are the nearest neighbors of x as defined by the span, and $d(x)$ is the distance along the abscissa from x to the most distant predictor value within the span.

The weights have these characteristics:

- The data point to be smoothed has the largest weight and the most influence on the fit.
- Data points outside the span have zero weight and no influence on the fit.

2. A weighted linear least-squares regression is performed. For lowess, the regression uses a first degree polynomial. For loess, the regression uses a second degree polynomial.
3. The smoothed value is given by the weighted regression at the predictor value of interest.

If the smooth calculation involves the same number of neighboring data points on either side of the smoothed data point, the weight function is symmetric. However, if the number of neighboring points is not symmetric about the smoothed data point, then the weight function is not symmetric. Note that unlike the moving average smoothing process, the span never changes. For example, when you smooth the data point with the smallest predictor value, the shape of the weight function is truncated by one half, the leftmost data point in the span has the largest weight, and all the neighboring points are to the right of the smoothed value.

The weight function for an end point and for an interior point is shown below for a span of 31 data points.

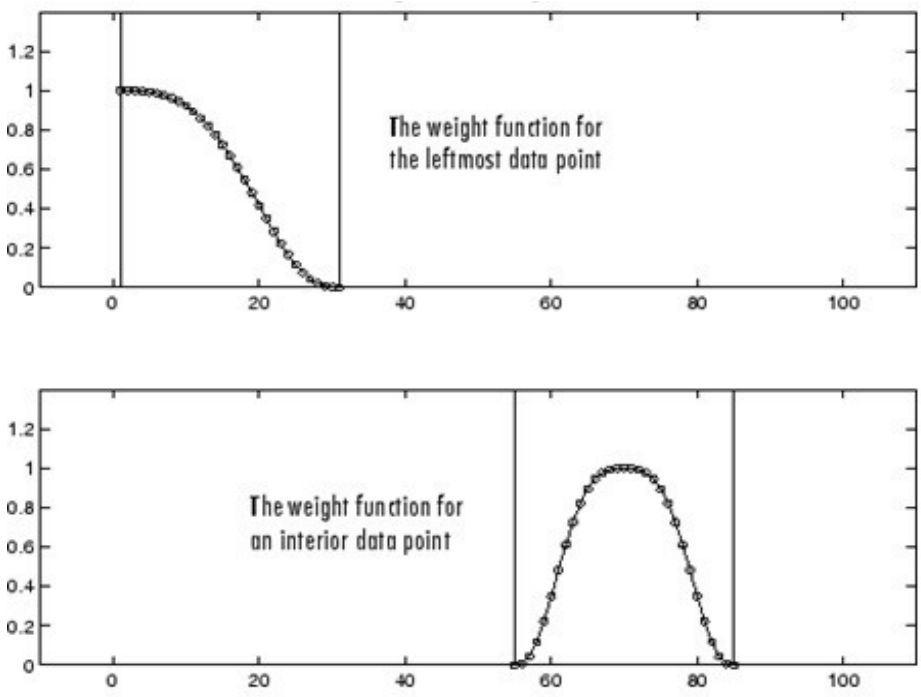

Figure 7.1: Local Regression Weight Function

Using the lowess method with a span of five, the smoothed values and associated regressions for the first four data points of a generated data set are shown below.

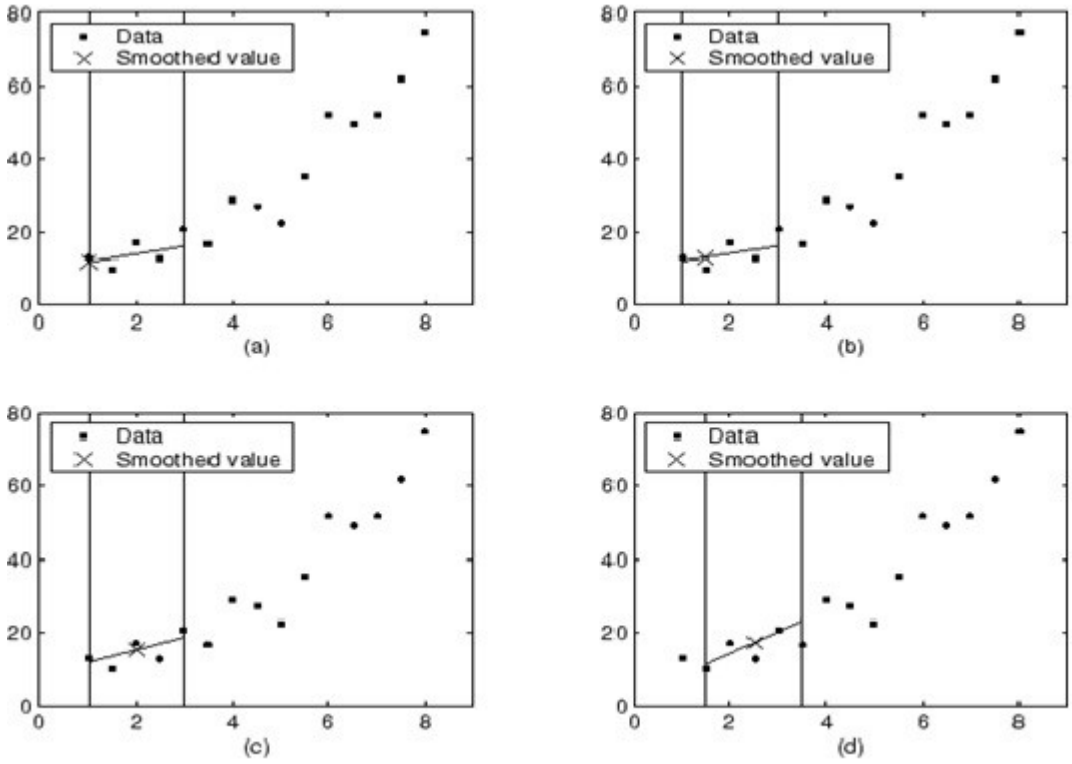

Figure 7.2: Lowess Smoothing

Notice that the span does not change as the smoothing process progresses from data point to data point. However, depending on the number of nearest neighbors, the regression weight function might not be symmetric about the data point to be smoothed. In particular, plots (a) and (b) use an asymmetric weight function, while plots (c) and (d) use a symmetric weight function.

For the loess method, the graphs would look the same except the smoothed value would be generated by a second-degree polynomial.

7.6.3 Robust Local Regression

If your data contains outliers, the smoothed values can become distorted, and not reflect the behavior of the bulk of the neighboring data points. To overcome this problem, you can smooth the data using a robust procedure that is not influenced by a small fraction of outliers. Curve Fitting Toolbox™ software provides a robust version for both the lowess and loess smoothing methods. These robust methods include an additional calculation of robust weights, which is resistant to outliers. The robust smoothing procedure follows these steps:

1. Calculate the residuals from the smoothing procedure described in the previous section.

2. Compute the *robust weights* for each data point in the span. The weights are given by the bisquare function,

$$w_i = \begin{cases} \left(1 - (r_i/6MAD)^2\right)^2, & |ri| < 6MAD, \\ 0, & |ri| \geq 6MAD, \end{cases}$$

where r_i is the residual of the ith data point produced by the regression smoothing procedure, and MAD is the median absolute deviation of the residuals, MAD=median(|r|)

The median absolute deviation is a measure of how spread out the residuals are. If r_i is small compared to 6MAD, then the robust weight is close to 1. If r_i is greater than 6MAD, the robust weight is 0 and the associated data point is excluded from the smooth calculation.

1.Smooth the data again using the robust weights. The final smoothed value is calculated using both the local regression weight and the robust weight.

2.Repeat the previous two steps for a total of five iterations.

The smoothing results of the lowess procedure are compared below to the results of the robust lowess procedure for a generated data set that contains a single outlier. The span for both procedures is 11 data points.

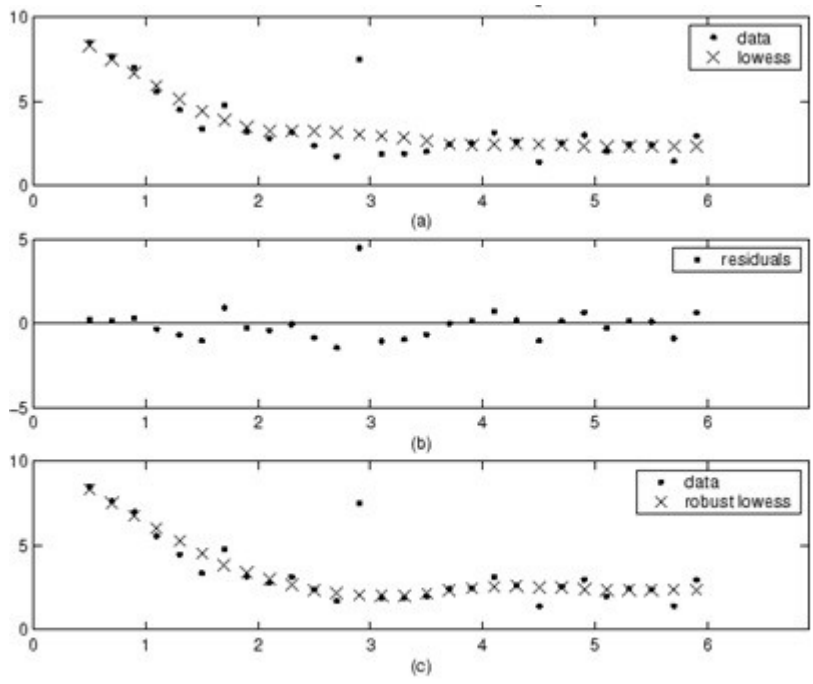

Figure 7.3: Loess Smoothing

Plot (a) shows that the outlier influences the smoothed value for several nearest neighbors. Plot (b) suggests that the residual of the outlier is greater than six median absolute deviations. Therefore, the robust weight is zero for this data point. Plot (c) shows that the smoothed values neighboring the outlier reflect the bulk of the data.

7.7 Normalized cross-correlation

The correlation between two signals (cross correlation) is a standard approach to feature detection.

The *correlation coefficient* normalizes the feature vectors to unit length, yielding a cosine-like correlation coefficient

$$\gamma(u,v) = \frac{\sum_{x,y}[f(x,y) - \bar{f}_{u,v}][t(x-u,y-v) - \bar{t}]}{\left\{\sum_{x,y}[f(x,y) - \bar{f}_{u,v}]^2 \sum_{x,y}[t(x-u,y-v) - \bar{t}]^2\right\}^{0.5}}$$

where $\bar{t}$ is the mean of the feature and $\bar{f}_{u,v}$ is the mean of $f(x,y)$ in the region under the feature. We refer this as *normalized cross-correlation*.

CHAPTER 8

EXPERIMENT RESULT/ CONCLUSION

This is a graph for a short video that had a suspense component in it. The falling slope indicates suspense emotion. The video had a fear component too. This is indicated by a sudden rise in the graph.

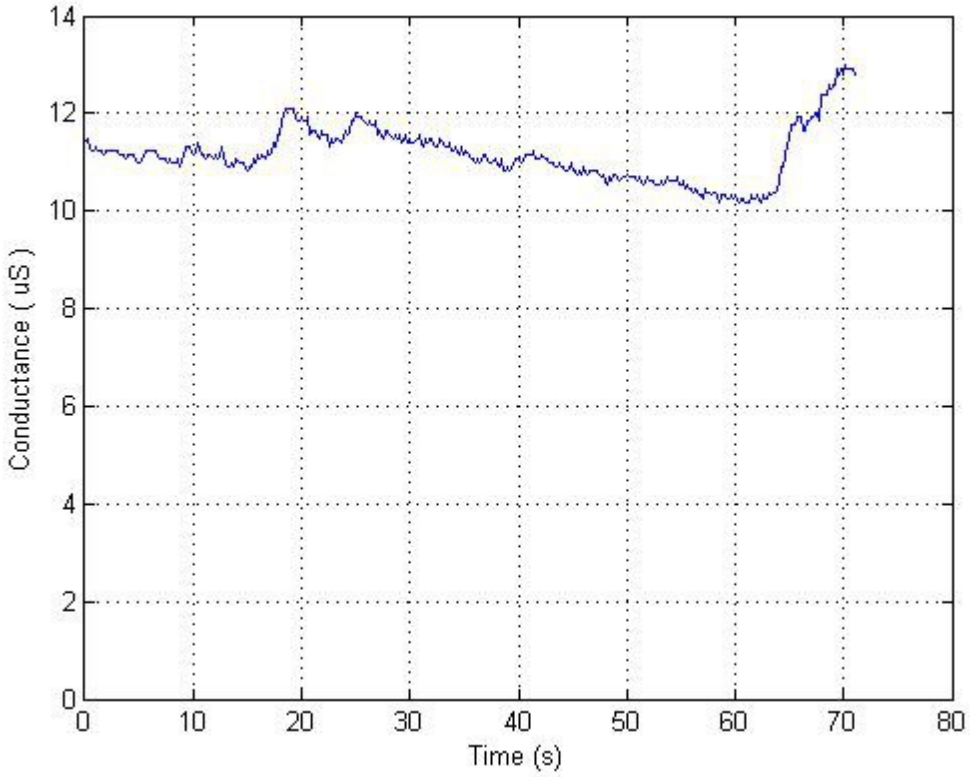

Figure 8.1: Input Graph

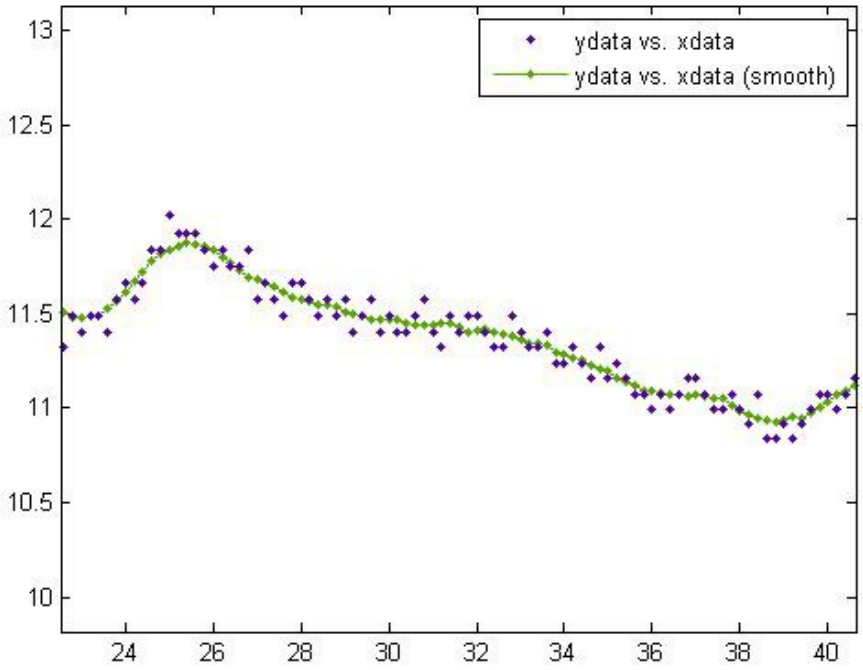

Figure 8.2: Smoothened Graph Using Moving Average Filter with a span of 9

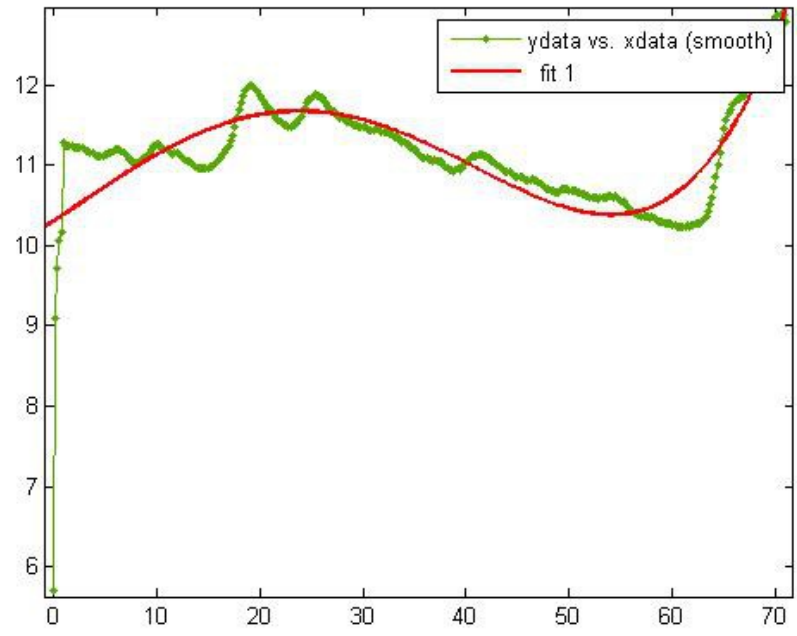

Figure 8.3: Curve Fitting Analysis

CHAPTER 8

FUTURE SCOPE

As we observe, the graphs obtained in these experiments with The Galvanic Skin Response are based on different emotions felt by the human body. Every emotion basically inflicts a certain involuntary change in the amount of sweat released which generates different patterns in the Skin Response.

The basic aim of this project was to find a way to detect the emotion of an individual through the process of pattern recognition in the form of time graphs. The analysis has been done through the observations in the pattern. In future, this project can be implemented with the other emotion recognition processes in artificial intelligence systems.

REFERENCES

1. M. Villarejo and B. Zapirain, “A Stress Sensor Based on Galvanic Skin Response (GSR) Controlled by Zigbee”, Sensor, Basel, 2012.
2. Shangguan G. Liu and W. Wen, “The Emotion Recognition Based on GSR Signal by Curve Fitting”, Journal of Information & Computational Science, vol. 2635-2646, 2014..

3. “Experiment HP_2: The Galvanic Skin Response (GSR) and Emotion”, in Human Psychophysiology “GSR-A” Lab , 1st ed., iWorx Systems Inc., 2015.
4. ANDREWS, “SKIN RESISTANCE CHANGES AND MEASUREMENT OF PAIN THRESSHOLD”, United States Public Health Service, United States Public Health Service Hospital, Lexington, Kentucky, 1942.
5. “Emotions States Recognition Based on Psycological Parameters by Employing of Fuzzy-Adaptive Resonance Theory”,International Journal of Intelligence Science, 2012, vol. 166-175, 2012.
6. Peuscher, “Galvanic skin response(GSR)”, 2012.
7. “WHY YOUR PALMS GET SWEATY WHEN YOU’RE EXITED, SCARED OR NERVOUR”, 2013.
8. Mathworks.com, “Filtering and Smoothing Data”, 2015. [Online]. Available: hhttp://in.mathworks.com/help/curvefit/smoothing-data.html?requestedDomain=in.mathworks.com.
9. Mathworks.com, “Signal Smoothing”, 2015. [Online]. Available: http://in.mathworks.com/help/signal/examples/signal-smoothing.html.
10. Wordpress.com, “Skin Response Meter” Viva Questions, 2015. [Online]. Available: https://dmohankumar.files.wordpress.com/2010/05/skin-response-meter-viva-questions.pdf.