

STRESS MONITORING SYSTEM USING FACIAL AND VOCAL RECOGNITION

A Project Report Phase-II

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Electronics and
Communication Engineering

By

39130532 JONNAKUTI VIJAYKUMAR

39130535 KALIGITHI TANAI



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SCHOOL OF ELECTRICAL AND ELECTRONICS

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600119

APRIL 2023



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC

BONAFIDE CERTIFICATE

This is to certify that this Project is the bonafide work of **J VIJAYKUMAR (39130532)** and **K.TANAI (39130535)** who Carried out the project entitled **"Stress monitoring system using facial and vocal recognition"** Under supervision for December 2022 to March 2023.

(Dr A. SAHAYA ANSELIN NISHA M.E.,Ph.D)

Internal Guide

Dr.T.RAVI,M.E.,Ph.D.,

Head of the Department

Submitted for Viva Voice Examination held on 20/04/2023

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We **J VIJAYKUMAR (39130532)** and **K TANAI (39130535)** hereby Declare that the Project Report entitled “**Stress monitoring system using facial and vocal recognition**” done by us under the guidance of **Dr.A.SHAYA ANESLIN NISHA** at Sathyabama Institute of Science and Technology, Chennai-600119 is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

DATE: 20/04/2023

PLACE: CHENNAI

1. 

2. 

SIGNATURE OF CANDIDATES

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. N.M. NANDHITHA, M.E., Ph.D., Dean**, School of Electrical and Electronics, **Dr. T. RAVI, M.E., Ph.D., Head of the Department, Dept. of Electronics and Communication Engineering** for providing us necessary support and details at the right time during the progressive reviews.

We would also like to express our sincere and deep sense of gratitude to our Project guide **Dr.A.SAHAYA ANESLIN NISHA**, Dept of Electronics & communication engineering for her valuable guidance and suggestions and constant encouragement during the progressive reviews for our project.

We wish to express my thanks to all Teaching and Non-teaching staff members of the Department of **Electronics and Communications engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Stress is a common experience for many people and can have significant negative impacts on mental and physical health. To address this issue, researchers have developed a stress monitoring system using facial and vocal recognition technology. This system uses advanced machine learning algorithms to analyse a person's facial expressions and vocal tone to detect signs of stress. The system then provides real-time feedback and recommendations to help individuals manage their stress levels. This technology has the potential to revolutionize stress management, providing people with a proactive tool to better understand and manage their stress levels. Additionally, the system could be integrated into various settings, such as healthcare, workplaces, and schools, to improve overall well-being and performance. However, there are also concerns about privacy and potential biases in the algorithm, which must be carefully addressed in the development and implementation of such systems. It is impossible to deny that students experience psychological stress in this digital and modern educational environment. The excessive accumulation of stress may result in a variety of issues, including a decrease in student grade (performance), an increase in violent behaviour, and even more severe cases.

Innovations that facilitate interactions between humans and things were made possible by the development of ICT (information and communication technology) and its tools. A novel, IoT-aware student-centric stress monitoring and real-time alert generation framework is proposed in this application to predict the student stress index in a specific context. To elaborate, we generated the scores of emotions from student facial expressions, speech pitch, and speech content at the cloud layer using extended. Specifically, the model aims to classify stress-related events as normal or abnormal based on the students' physiological data readings' overall emotion. The enactment of the strange occasion in the event of higher qualities for pessimistic feelings like pressure, dread, trouble, disdain, and so forth. The student, coordinators, and caretakers receive a firm warning. In the end, the framework that has been proposed will be a great tool that will help education institutions, students, and their parents and guardians monitor students' overall emotions in real time.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	viii
1	INTRODUCTION	1
	1.1 Proposed Algorithm	1
2	LITERATURE SURVEY	7
	2.1 Inferences from Literature Survey	8
3	SYSTEM REQUIRMENTS	12
	3.1 Hardware Requirements	12
	3.2 Software Requirements	12
	3.3 Language Specification	12
	3.4 History of Python	12
	3.5 Application of Python	12
	3.6 Features of Python	13
	3.7Feasibility Study	14
4	SYSTEM ANALYSIS	15
	4.1 Purpose	15
	4.2 Scope	15
	4.3 Existing System	15
	4.4 Proposed System	16
5	SYSTEM DESIGN & IMAGEPROCESSING	43
6	SYSTEM IMPLEMENTATION	44
	6.1 System Architecture	44
7	SYSTEM TESTING	45
	7.1Test Plan	45

	7.2 Verification	45
	7.3 Validation	45
	7.4 Basic of Software Testing	45
	7.5 Black Box Testing	45
	7.6 White Box Testing	45
	7.7 Types of Testing	46
8	RESULT AND DISCUSSION	50
9	CONCLUSION	54
	REFERENCES	55

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
1.1	Classification of methods	2
1.2	Filter matrix	3
1.3	Image and filter matrix	3
1.4	Convolved feature	3
1.5	Strides	3
1.6	Padding	4
1.7	Maxpooling	5
1.8	Fully connected layer	6
5.1	Flow chart	18
5.2	LUT	23
5.3	Convolution	23
5.4	illustration graph	27
5.5	Convolution kernel for a mean filter	31
5.6	Truth tables for AND & NAND	32
5.7	Simple gray level image&corresponding surface	36
5.8	Two Connected components based on 4-connectivity	40
5.9	Some examples structuring elements	43
6.1	System architecture	44
8.1.1	Happy	50
8.1.2	Neutral	50
8.1.3	Sad	50
8.1.4	Angry	51
8.1.5	Surprise	51
8.2.1	Import data	51
8.2.2	Pre-processed data	52
8.2.3	Trained data	52
8.2.4	Graph1	52
8.2.5	Graph2	53

CHAPTER 1

INTRODUCTION

The pedagogy of education teaching and learning has developed over time. Information and communication technology (ICT) and other related fields' technological advancements have a direct impact on education worldwide. It has been observed that in the modern era of technology, both teachers and students relied excessively on technology for the purpose of achieving the goals of successful learning. This infusion of technology with education learning and teaching pedagogy, or education theory, has resulted in the development of a new implication in education known as education technology. Besides, the presentation of contemporary showing instructional methods like mixed learning and flipped learning (homerooms) in schooling has uncovered powerful proof of moderate improvements in the space of schooling. With the declarations of instruction educating and learning movement and improvement, the presence of unfriendly results and their age because of different extraneous and natural elements can't be denied.

1.1 PROPOSED ALGORITHMS

Convolutional Neural Network(CNN Algorithm) is one of the main categories to do image classification and image recognition in neural networks. Scene labeling, objects detections, and face recognition, etc., are some of the areas where convolutional neural networks are widely used.

CNN takes an image as input, which is classified and process under a certain category such as dog, cat, lion, tiger, etc. The computer sees an image as an array of pixels and depends on the resolution of the image. Based on image resolution, it will see as $h * w * d$, where h = height w = width and d = dimension. For example, An RGB image is $6 * 6 * 3$ array of the matrix, and the grayscale image is $4 * 4 * 1$ array of the matrix.

In CNN, each input image will pass through a sequence of convolution layers along with pooling, fully connected layers, filters (Also known as kernels). After that, we will apply the Soft-max function to classify an object with probabilistic values 0 and 1.

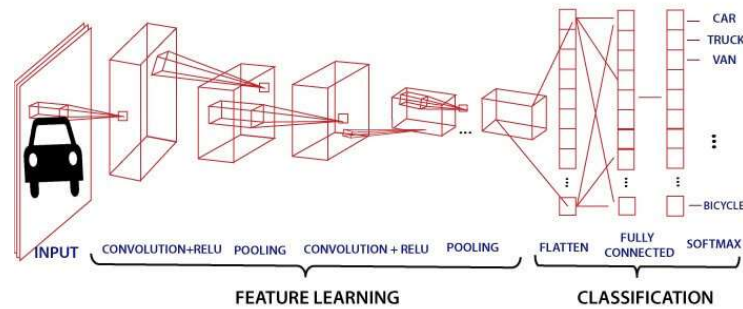


Fig 1.1 Classification of methods

Convolution layer is the first layer to extract features from an input image. By learning image features using a small square of input data, the convolutional layer preserves the relationship between pixels. It is a mathematical operation which takes two inputs such as image matrix and a kernel or filter.

- The dimension of the image matrix is $h \times w \times d$.
- The dimension of the filter is $f_h \times f_w \times d$.
- The dimension of the output is $(h-f_h+1) \times (w-f_w+1) \times 1$.

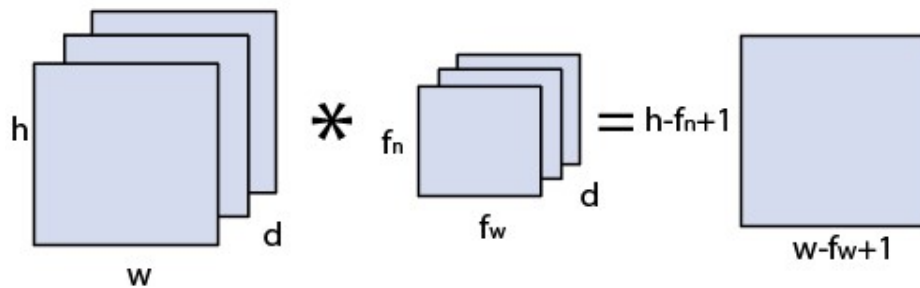


Fig1.2 Filter matrix

Let's start with consideration a 5×5 image whose pixel values are 0, 1, and filter matrix 3×3 as:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Fig1.3 Image and filter Matrix

The convolution of 5*5 image matrix multiplies with 3*3 filter matrix is called "**Features Map**" and show as an output.

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 3 & 4 \\ 2 & 4 & 3 \\ 2 & 3 & 4 \end{bmatrix}$$

Fig1.4 Convolved Feature

Convolution of an image with different filters can perform an operation such as blur, sharpen, and edge detection by applying filters.

Stride is the number of pixels which are shift over the input matrix. When the stride is equalled to 1, then we move the filters to 1 pixel at a time and similarly, if the stride is equalled to 2, then we move the filters to 2 pixels at a time. The following figure shows that the convolution would work with a stride of 2.

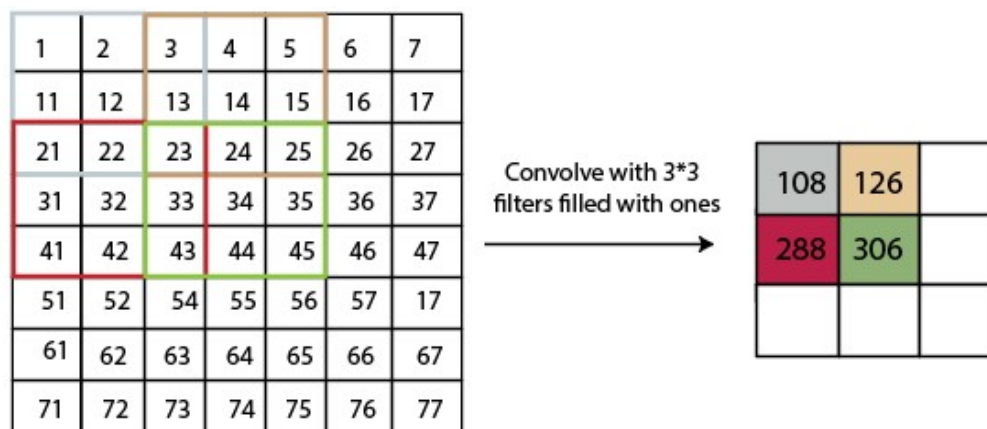


Fig1.5 Strides

Padding plays a crucial role in building the convolutional neural network. If the image will get shrink and if we will take a neural network with 100's of layers on it, it will give us a small image after filtered in the end. If we take a three by three filter on top of a grayscale image and do the convolving then what will happen?

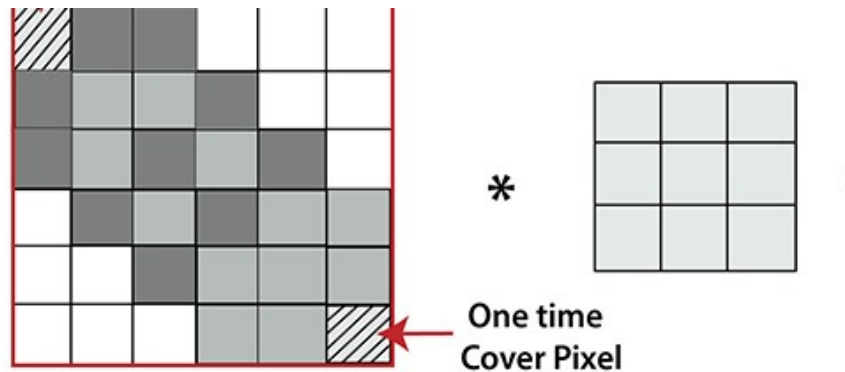


Fig1.6 padding

It is clear from the above picture that the pixel in the corner will only get covered one time, but the middle pixel will get covered more than once. It means that we have more information on that middle pixel, so there are two downsides:

- Shrinking outputs
- Losing information on the corner of the image.
- To overcome this, we have introduced padding to an image. **"Padding is an additional layer which can add to the border of an image."**

Pooling layer plays an important role in pre-processing of an image. Pooling layer reduces the number of parameters when the images are too large. Pooling is "**downscaling**" of the image obtained from the previous layers. It can be compared to shrinking an image to reduce its pixel density. Spatial pooling is also called down sampling or subsampling, which reduces the dimensionality of each map but retains the important information.

There are the following types of spatial pooling:

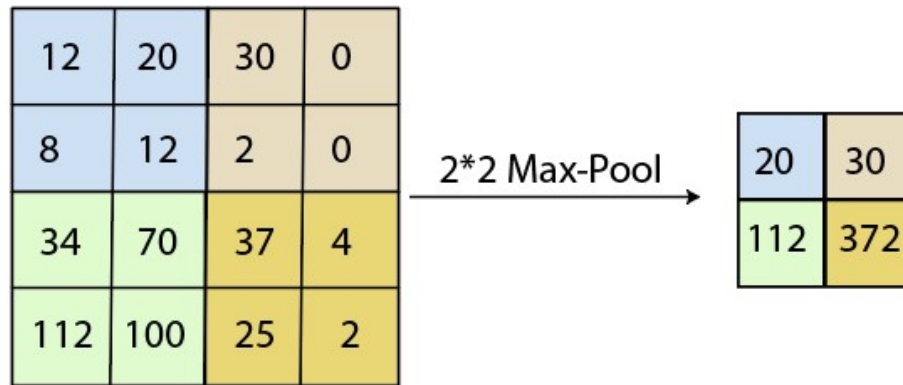


Fig1.7 Maxpooling

Down-scaling will perform through average pooling by dividing the input into rectangular pooling regions and computing the average values of each region.

`layer = averagePooling2dLayer(poolSize)`

`layer = averagePooling2dLayer(poolSize,Name,Value)`

Sum Pooling is the sub-region for **sum pooling** or **mean pooling** are set exactly the same as for **max-pooling** but instead of using the max function we use sum or mean. The fully connected layer is a layer in which the input from the other layers will be flattened into a vector and sent. It will transform the output into the desired number of classes by the network.

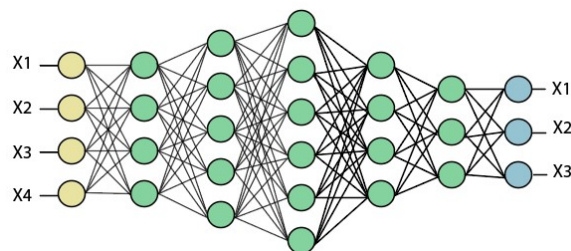


Fig 1.8 Fully Connected Layer

In the above diagram, the feature map matrix will be converted into the vector such as **x1, x2, x3... xn** with the help of fully connected layers. We will combine features to create a model and apply the activation function such as **softmax** or **sigmoid** to classify the outputs as a car, dog, truck, etc.

CHAPTER 2

LITERATURE SURVEY

2.1 Inferences from Literature Survey

Kaulard [26] stated that One of the most important aspects of human existence is communication. We employ remarkable complexity of both verbal and nonverbal signals for this. Facial expressions are one of the most important information channels among the latter. Thus, its standing the enormous assortment of looks we use in day to day existence, research on looks has up until this point generally centered around the close to home viewpoint. As a result, the majority of facial expression databases that are accessible to researchers also only include emotional expressions and ignore the largely unexplored aspect of conversational expressions. The MPI facial expression database, which includes a wide range of natural emotional and conversational expressions, fills this void. 19 German participants performed 55 distinct facial expressions in the database. Articulations were inspired with the assistance of a strategy acting convention, which ensures both distinct and regular looks. Everyday scenarios served as the foundation for the method-acting protocol, which was used to define the necessary context information for each expression. Every facial expression can be seen three times, in two different intensities, and from three different camera angles. A dynamic and a static version of the database have been created from a detailed annotation of the frame. In addition to providing a comprehensive description of the database, we also present the outcomes of an experiment conducted under two conditions to validate the context scenarios as well as the video sequences' naturalness and recognizability. Based on our findings, it is clear that visual information alone is sufficient to recognize conversational expressions. The MPI facial expression database will make it possible for researchers in a variety of fields to investigate the processing of a wider variety of natural facial expressions. These fields include the perceptual and cognitive sciences, affective computing, and computer vision.

G.E. Hinton [27] dealt with the temporal variability of speech, the majority of current speech recognition systems employ hidden Markov models (HMMs) and Gaussian mixture models (GMMs) to determine how well each state of each HMM

fits a frame or a brief window of frames of coefficients that represent the acoustic input. Using a feed-forward neural network, which takes several frames of coefficients as input and generates posterior probabilities over HMM states as output, is an alternative method for evaluating the fit. On a variety of speech recognition benchmarks, it has been demonstrated that deep neural networks (DNNs), which are trained using novel techniques and have a lot of hidden layers, perform better than GMMs, sometimes by a lot. The viewpoints of four research groups that have recently achieved success in utilizing DNNs for acoustic modelling in speech recognition are presented in this article, which provides an overview of these advancements.

Pentland [28] stated that Words alone cannot convey the social context of face-to-face communication. This social signaling facilitates the seamless integration of new information into a collective, group-wide understanding. Signals of interest, determination, friendliness, boredom, and other "attitudes" toward a social situation are all examples of social signaling. Due to the fact that social signaling functions as a subconscious discussion about relationships, resources, risks, and rewards, psychologists speculate that social signaling may have evolved as a means of establishing hierarchy and group cohesion. When it comes to determining human behavior, the non-linguistic signals that form the basis of this social discussion are frequently just as significant as conscious content. The difficulties associated with exploratory processing of social signals as well as the tools that enable us to predict human behavior and sometimes surpass even expert human capabilities are discussed in the following sections. Instead of treating the individual as an isolated entity, computer and communications systems may be able to support social and organizational roles thanks to these tools. with numerous applications and challenges. While significant progress has been made in recent years, there are still many opportunities for further research and development. As such, object detection is likely to remain an active area of research and innovation in the coming years, with the potential to transform a wide range of industries and domains. Automatically patching people into socially significant conversations, initiating conversations among people to build a stronger social network, and strengthening family ties are examples of applications.

M. Xie [29] taught about the rapid development of artificial intelligence (AI) and machine learning, their application has been widely used in many areas of the financial sector in recent years and has had significant effects on the financial market, institutions, and regulation. The advancement of artificial intelligence technology significantly alters the financial sector as a whole, leading to the development of a number of cutting-edge financial services like intelligent consulting, intelligent lending, monitoring and warning, and intelligent customer service in response to changing circumstances. The purpose is to provide a summary of the financial system's use of artificial intelligence and machine learning, as well as their effects on macroeconomics and microeconomics. In the meantime, it is becoming clear that artificial intelligence caused a number of issues and risks during its use. Last but not least, based on the financial risk management issues posed by artificial intelligence, some ideas and tactics for using artificial intelligence in a reasonable way in financial risk management are provided.

A. Nandi [30] experimented about Emotions and emotional intelligence are important aspects of both in-person and online education. Students' feelings are critical for e-learning framework since they advance or control the learning. Emotions have been the subject of numerous studies aimed at improving and maximizing e-learning outcomes. To accomplish this, a number of machine learning and deep learning strategies have also been proposed. All of these methods work well in offline mode, where the data used to classify emotions are stored and accessible indefinitely. However, when data are being fed into the model in a continuous stream and can only be seen by the model at once, offline mode methods are not appropriate for real-time emotion classification. We also need emotional state-specific responses in real time. We propose a logistic regression (LR) based on the real-time emotion classification system (RECS), which is trained online with the Stochastic Gradient Descent (SGD) algorithm. The proposed RECS is equipped for ordering feelings progressively via preparing the model in a web-based style utilizing an EEG signal stream. We used the DEAP data set, which is the most widely used benchmark data set for emotion classification, to verify RECS's performance. The findings demonstrate that the proposed method is capable of accurately and with a higher F1-score than other offline and online methods of classifying emotions in real time from the EEG data stream. In an e-learning context, the developed real-time

emotion classification system is examined.

A. Raheel [31] proposed importance of Emotion recognition has increased the potential of affective computing by getting instant feedback from users and thereby, have a better understanding of their behavior. Physiological sensors have been used to recognize human emotions in response to audio and video content that engages single (auditory) and multiple (two: auditory and vision) human senses, respectively. In this study, human emotions were recognized using physiological signals observed in response to tactile enhanced multimedia content that engages three (tactile, vision, and auditory) human senses. The aim was to give users an enhanced real-world sensation while engaging with multimedia content. To this end, four videos were selected and synchronized with an electric fan and a heater, based on timestamps within the scenes, to generate tactile enhanced content with cold and hot air effect respectively. Physiological signals, i.e., electroencephalography (EEG), photo plethysmography (PPG), and galvanic skin response (GSR) were recorded using commercially available sensors, while experiencing these tactile enhanced videos. The precision of the acquired physiological signals (including EEG, PPG, and GSR) is enhanced using pre-processing with a Savitzky- Golay smoothing filter. Frequency domain features (rational asymmetry, differential asymmetry, and correlation) from EEG, time domain features (variance, entropy, kurtosis, and skewness) from GSR, heart rate and heart rate variability from PPG data are extracted. The K nearest neighbor classifier is applied to the extracted features to classify four (happy, relaxed, angry, and sad) emotions. Our experimental results show that among individual modalities, PPG-based features gives the highest accuracy of 78.57% as compared to EEG- and GSR-based features. The fusion of EEG, GSR, and PPG features further improved the classification accuracy to 79.76% (for four emotions) when interacting with tactile enhanced multimedia.

D. Keltiner [32] Researched into emotions was just beginning to take off when the first edition of the Handbook of Emotions was published in 1993. Since then, it has been acknowledged as a crucial component of any study of humanity. The importance of studying emotions in relation to cognition, personality, social problems, and health issues has become clear to everyone since the turn of the century. By allowing us to consider the work that had already been completed and to begin exploring the new work that lay ahead, the second edition of the Handbook

attempted to meet this new challenge. The principal version was appraised by Decision as an "Extraordinary Scholastic Book" in 1995. The third release proceeds with this practice, as it tries to be both a summary of what we have learned and a prologue to new regions that have gotten our creative mind. This volume should pique the interest of those just getting started in the field of emotion research by providing them with new areas to investigate and fresh perspectives on previously unanswered questions. For we who keep on working in the field and who once toiled as detached researchers, this third version shows the strength and imperativeness of the field as new regions open up and new agents make outstanding commitments. This development shows how many academics are now interested in emotional research and conceptual questions.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

System	: Pentium i3 Processor
Hard Disk	: 500 GB.
Monitor	: 15" LED
Input Devices	: Keyboard, Mouse
Ram	: 2 GB

3.2 SOFTWARE REQUIREMENTS

Operating system	: Windows 10
Coding Language	: Python

3.3 LANGUAGE SPECIFICATION

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This **tutorial** gives enough understanding on **Python programming** language.

3.4. HISTORY OF PYTHON

- Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.
- Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and Unix shell and other scripting languages.
- Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).
- Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

3.5. APPLICATION OF PYTHON

- **Easy-to-learn** – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** – Python code is more clearly defined and visible to the eyes.

- **Easy-to-maintain** – Python's source code is fairly easy-to-maintain.
- **A broad standard library** – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** – Python provides interfaces to all major commercial databases.
- **GUI Programming** – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** – Python provides a better structure and support for large programs than shell scripting.

3.6 FEATURES OF PYTHON

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

3.7 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility

analysis, some understanding of the major requirements for the system is essential. The feasibility study investigates the problem and the information needs of the stakeholders. It seeks to determine the resources required to provide an information systems solution, the cost and benefits of such a solution, and the feasibility of such a solution.

The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most suitable to the organization. The feasibility of a proposed solution is evaluated in terms of its components.

3.7.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system is well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.7.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.7.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

CHAPTER 4

SYSTEM ANALYSIS

4.1 PURPOSE

The purpose of this document is real time facial expression based music recommender system using machine learning algorithms. In detail, this document will provide a general description of our project, including user requirements, product perspective, and overview of requirements, general constraints. In addition, it will also provide the specific requirements and functionality needed for this project - such as interface, functional requirements and performance requirements.

4.2 SCOPE

The scope of this SRSdocument persists for the entire life cycle of the project. This document defines the final state of the software requirements agreed upon by the customers and designers. Finally at the end of the project execution all the functionalities may be traceable from the SRS to the product. The document describes the functionality, performance, constraints, interface and reliability for the entire life cycle of the project.

4.3 EXISTING SYSTEM

- In the existing system we are used different technologies to identify the stress Stress has emerged as one of the main causes of health issues as a result of the increasing pace of life.
- We have created a method for continuously and unobtrusively measuring stress in real-world situations. We present stress data derived from sensor measurements in relation to a person's activities in order to provide information that is meaningful, useful, and actionable.
- In this paper, we talk about our framework, how we deal with new problems, and how we think about our approach based on the field studies we did.
- The principal consequences of the assessment are that the consequences of long-haul estimations of stress uncover individuals data about their personal conduct standards that they see as significant and valuable, and trigger their thoughts regarding social changes important to accomplish a superior pressure balance.

4.4 PROPOSED SYSTEM

- In the proposed system we are going to use deep learning algorithm to find out the stress through face emotions and vocal.
- While using face emotions we are going to find out the five emotions like angry, disgust, happy, fear and sad.
- The process is going to happen with the help of datasets collected from Kaggle we are going to find out the emotions through web cam
- Coming to vocal emotion recognition we are going to test and train the datasets which is full of voice note and finally we are going to find out the actual and detected one.

CHAPTER 5

SYSTEM DESIGN & IMAGE PROCESSING

Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image. Usually, **Image Processing** system includes treating images as two-dimensional signals while applying already set signal processing methods to them.

It is among rapidly growing technologies today, with its applications in various aspects of a business. Image Processing forms core research area within engineering and computer science disciplines too.

Image processing basically includes the following three steps.

- Importing the image with optical scanner or by digital photography.
- Analyzing and manipulating the image which includes data compression and image enhancement and spotting patterns that are not to human eyes like satellite photographs.
- Output is the last stage in which result can be altered image or report that is based on image analysis.

The purpose of image processing is divided into 5 groups. They are:

1. Visualization - Observe the objects that are not visible.
2. Image sharpening and restoration - To create a better image.
3. Image retrieval - Seek for the image of interest.
4. Measurement of pattern – Measures various objects in an image.
5. Image Recognition – Distinguish the objects in an image.

The two types of **methods used for Image Processing** are **Analog and Digital** Image Processing. Analog or visual techniques of image processing can be used for the hard copies like printouts and photographs. Image analysts use various fundamentals of interpretation while using these visual techniques. The image processing is not just confined to area that has to be studied but on knowledge of analyst. Association is another important tool in image processing through visual techniques. So, analysts apply a combination of personal knowledge and collateral data to image processing. Digital Processing techniques help in manipulation of the digital images by using computers. As raw data from imaging sensors from satellite

platform contains deficiencies. To get over such flaws and to get originality of information, it has to undergo various phases of processing. The three general phases that all types of data have to undergo while using digital technique are Pre-processing, enhancement and display, information extraction.

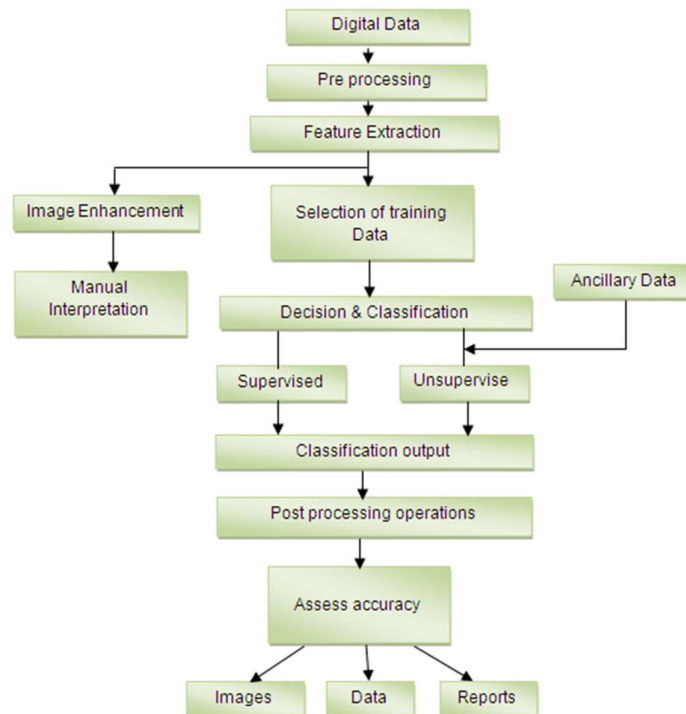


Fig5.1flow chart

Binary images are images whose pixels have only two possible intensity values. They are normally displayed as black and white. Numerically, the two values are often 0 for black, and either 1 or 255 for white.

Binary images are often produced by thresholding a grayscale or color image, in order to separate an object in the image from the background. The color of the object (usually white) is referred to as the foreground color. The rest (usually black) is referred to as the background color. However, depending on the image which is to be thresholded, this polarity might be inverted, in which case the object is displayed with 0 and the background is with a non-zero value. Some morphological operators assume a certain polarity of the binary input image so that if we process an image with inverse polarity the operator will have the opposite effect. For example, if we apply a closing operator to a black text on white background, the text will be opened.

It is possible to construct (almost) all visible colors by combining the three primary colors red, green and blue, because the human eye has only three different color

receptors, each of them sensible to one of the three colors. Different combinations in the stimulation of the receptors enable the human eye to distinguish approximately 350000 colors. A RGB color image is a multi-spectral image with one band for each color red, green and blue, thus producing a weighted combination of the three primary colors for each pixel. A full 24-bit color image contains one 8-bit value for each color, thus being able to display different colors.

However, it is computationally expensive and often not necessary to use the full 24-bit image to store the color for each pixel. Therefore, the color for each pixel is often encoded in a single byte, resulting in an 8-bit color image. The process of reducing the color representation from 24-bits to 8-bits, known as color quantization, restricts the number of possible colors to 256. However, there is normally no visible difference between a 24-bit color image and the same image displayed with 8 bits. An 8-bit color images are based on colormaps, which are look-up tables taking the 8-bit pixel value as index and providing an output value for each color.

Full RGB color requires that the intensities of three color components be specified for each and every pixel. It is common for each component intensity to be stored as an 8-bit integer, and so each pixel requires 24 bits to completely and accurately specify its color. If this is done, then the image is known as a 24-bit color image. However, there are two problems with this approach. Storing 24 bits for every pixel leads to very large image files that with current technology are cumbersome to store and manipulate. For instance, a 24-bit 512×512 image takes up 750KB in uncompressed form.

- Many monitor displays use colormaps with 8-bit index numbers, meaning that they can only display 256 different colors at any one time.
- Thus it is often wasteful to store more than 256 different colors in an image anyway, since it will not be possible to display them all on screen.

Because of this, many image formats (e.g. 8-bit GIF and TIFF) use 8-bit colormaps to restrict the maximum number of different colors to 256. Using this method, it is only necessary to store an 8-bit index into the colormap for each pixel, rather than the full 24-bit color value. Thus 8-bit image formats consist of two parts:

a colormap describing what colors are present in the image, and the array of index values for each pixel in the image.

When a 24-bit full color image is turned into an 8-bit image, it is usually necessary to throw away some of the colors, a process known as color quantization. This leads to some degradation in image quality, but in practice the observable effect can be quite small, and in any case, such degradation is inevitable if the image output device (e.g. screen or printer) is only capable of displaying 256 colors or less. The use of 8-bit images with colormaps does lead to some problems in image processing. First of all, each image has to have its own colormap, and there is usually no guarantee that each image will have exactly the same colormap. Thus on 8-bit displays it is frequently impossible to correctly display two different color images that have different colormaps at the same time. Note that in practice 8-bit images often use reduced size colormaps with less than 256 colors in order to avoid this problem.

Another problem occurs when the output image from an image processing operation contains different colors to the input image or images. This can occur very easily, as for instance when two color images are added together pixel-by-pixel. Since the output image contains different colors from the input images, it ideally needs a new colormap, different from those of the input images, and this involves further color quantization which will degrade the image quality. Hence the resulting output is usually only an approximation of the desired output. Repeated image processing operations will continually degrade the image colors. And of course we still have the problem that it is not possible to display the images simultaneously with each other on the same 8-bit display.

Because of these problems it is to be expected that as computer storage and processing power become cheaper, there will be a shift away from 8-bit images and towards full 24-bit image processing.

- Full RGB color requires that the intensities of three-color components be specified for each and every pixel. It is common for each component intensity to be stored as an 8-bit integer, and so each pixel requires 24 bits to completely and accurately specify its color. Image formats that store a full 24

bits to describe the color of each and every pixel are therefore known as 24-bit color images.

- Using 24 bits to encode color information allows different colors to be represented, and this is sufficient to cover the full range of human color perception fairly well.

The term 24-bit is also used to describe monitor displays that use 24 bits per pixel in their display memories, and which are hence capable of displaying a full range of colors. There are also some disadvantages to using 24-bit images. Perhaps the main one is that it requires three times as much memory, disk space and processing time to store and manipulate 24-bit color images as compared to 8-bit color images. In addition, there is often not much point in being able to store all those different colors if the final output device (e.g. screen or printer) can only actually produce a fraction of them. Since it is possible to use colormaps to produce 8-bit color images that look almost as good, at the time of writing 24-bit displays are relatively little used. However, it is to be expected that as the technology becomes Color quantization is applied when the color information of an image is to be reduced. The most common case is when a 24-bit color image is transformed into an 8-bit color image. Two decisions have to be made which colors of the larger color set remain in the new image, and how are the discarded colors mapped to the remaining ones.

The simplest way to transform a 24-bit color image into 8 bits is to assign 3 bits to red and green and 2 bits to blue (blue has only 2 bits, because of the eye's lower sensitivity to this color). This enables us to display 8 different shades of red and green and 4 of blue. However, this method can yield only poor results. For example, an image might contain different shades of blue which are all clustered around a certain value such that only one shade of blue is used in the 8-bit image and the remaining three blues are not used.

Alternatively, since 8-bit color images are displayed using a colormap, we can assign any arbitrary color to each of the 256 8-bit values and we can define a separate colormap for each image. This enables us perform a color quantization adjusted to the data contained in the image. One common approach is the popularity algorithm, which creates a histogram of all colors and retains the 256 most frequent

ones. Another approach, known as the median-cut algorithm, yields even better results but also needs more computation time. This technique recursively fits a box around all colors used in the color space which it splits at the median value of its longest side. The algorithm stops after 255 recursions. All colors in one box are mapped to the centroid of this box.

All above techniques restrict the number of displayed colors to 256. A technique of achieving additional colors is to apply a variation of half-toning used for gray scale images, thus increasing the color resolution at the cost of spatial resolution. The 256 values of the colormap are divided into four sections containing 64 different values of red, green, blue and white. As can be seen in Figure 1, a 2×2 pixel area is grouped together to represent one composite color, each of the four pixels displays either one of the primary colors or white. In this way, the number of possible colors is increased from 256 to 64^4 .

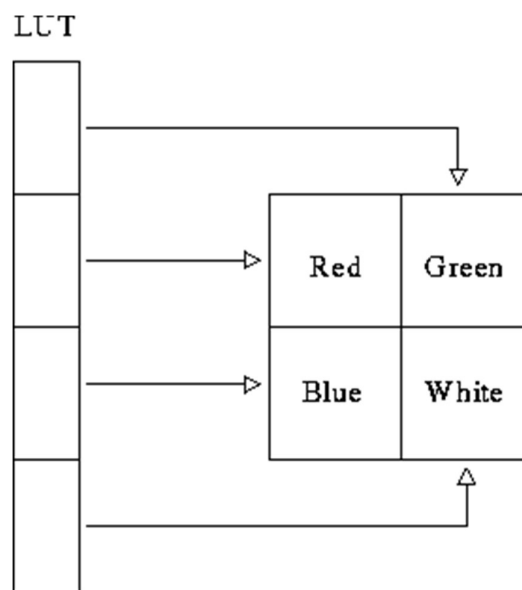


Fig 5.2 LUT

A 2×2 pixel area displaying one composite color cheaper, their use in image processing will grow. Convolution is a simple mathematical operation which is fundamental to many common image processing operators. Convolution provides a way of 'multiplying together' two arrays of numbers, generally of different sizes, but of the same dimensionality, to produce a third array of numbers of the same dimensionality. This can be used in image processing to implement operators whose output pixel values are simple linear combinations of certain input pixel values. In an

image processing context, one of the input arrays is normally just a graylevel image. The second array is usually much smaller, and is also two-dimensional (although it may be just a single pixel thick), and is known as the kernel. Figure 1 shows an example image and kernel that we will use to illustrate convolution.

I ₁₁	I ₁₂	I ₁₃	I ₁₄	I ₁₅	I ₁₆	I ₁₇	I ₁₈	I ₁₉
I ₂₁	I ₂₂	I ₂₃	I ₂₄	I ₂₅	I ₂₆	I ₂₇	I ₂₈	I ₂₉
I ₃₁	I ₃₂	I ₃₃	I ₃₄	I ₃₅	I ₃₆	I ₃₇	I ₃₈	I ₃₉
I ₄₁	I ₄₂	I ₄₃	I ₄₄	I ₄₅	I ₄₆	I ₄₇	I ₄₈	I ₄₉
I ₅₁	I ₅₂	I ₅₃	I ₅₄	I ₅₅	I ₅₆	I ₅₇	I ₅₈	I ₅₉
I ₆₁	I ₆₂	I ₆₃	I ₆₄	I ₆₅	I ₆₆	I ₆₇	I ₆₈	I ₆₉

K ₁₁	K ₁₂	K ₁₃
K ₂₁	K ₂₂	K ₂₃

Fig5.3convolution

An example small image (left) and kernel (right) to illustrate convolution. The labels within each grid square are used to identify each square. The convolution is performed by sliding the kernel over the image, generally starting at the top left corner, so as to move the kernel through all the positions where the kernel fits entirely within the boundaries of the image. (Note that implementations differ in what they do at the edges of images, as explained below.) Each kernel position corresponds to a single output pixel, the value of which is calculated by multiplying together the kernel value and the underlying image pixel value for each of the cells in the kernel, and then adding all these numbers together. If the image has M rows and N columns, and the kernel has m rows and n columns, then the size of the output image will have M - m + 1 rows, and N - n + 1 columns.

Note that many implementations of convolution produce a larger output image than this because they relax the constraint that the kernel can only be moved to positions where it fits entirely within the image. Instead, these implementations typically slide the kernel to all positions where just the top left corner of the kernel is within the image. Therefore, the kernel 'overlaps' the image on the bottom and right edges. One advantage of this approach is that the output image is the same size as the input image. Unfortunately, in order to calculate the output pixel values for the bottom and right edges of the image, it is necessary to invent input pixel values for places where the kernel extends off the end of the image. Typically, pixel values of zero are chosen for regions outside the true image, but this can often distort the output image at these places. Therefore, in general if you are using a convolution

implementation that does this, it is better to clip the image to remove these spurious regions. Removing $n - 1$ pixels from the right-hand side and $m - 1$ pixels from the bottom will fix things. Convolution can be used to implement many different operators, particularly spatial filters and feature detectors. Examples include Gaussian smoothing and the Sobel edge detector.

It is often useful in image processing to be able to calculate the distance between two pixels in an image, but this is not as straightforward as it seems. The presence of the pixel grid makes several so-called distance metrics possible which often give different answers to each other for the distance between the same pair of points. We consider the three most important ones.

Euclidean Distance This is the familiar straight-line distance that most people are familiar with. If the two pixels that we are considering have coordinates (x_1, y_1) and (x_2, y_2) , then the Euclidean distance is given by:

$$D_{Euclid} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \text{ --eq1}$$

City Block Distance Also known as the Manhattan distance. This metric assumes that in going from one pixel to the other it is only possible to travel directly along pixel grid lines. Diagonal moves are not allowed. Therefore the 'city block' distance is given by:

$$D_{City} = |x_2 - x_1| + |y_2 - y_1| \text{ --eq2}$$

Chessboard Distance This metric assumes that you can make moves on the pixel grid as if you were a King making moves in chess, *i.e.* a diagonal move counts the same as a horizontal move. This means that the metric is given by:

$$D_{Chess} = \max(|x_2 - x_1|, |y_2 - y_1|) \text{ --eq3}$$

Note that the last two metrics are usually much faster to compute than the Euclidean metric and so are sometimes used where speed is critical but accuracy is not too important.

Dithering is an image display technique that is useful for overcoming limited display resources. The word dither refers to a random or semi-random perturbation of the pixel values.

Two applications of this techniques are particularly useful:

🟡 Low quantization display: When images are quantized to a few bits (e.g. 3) then only a limited number of graylevels are used in the display of the image. If the scene is smoothly shaded, then the image display will generate rather distinct boundaries around the edges of image regions when the original scene intensity moves from one quantization level to the next. To eliminate this effect, one dithering technique adds random noise (with a small range of values) to the input signal before quantization into the output range. This randomizes the quantization of the pixels at the original quantization boundary, and thus pixels make a more gradual transition from neighborhoods containing 100% of the first quantization level to neighborhoods containing 100% of the second quantization level.

🟡 Limited color display: When fewer colors are able to be displayed (e.g. 256) than are present in the input image (e.g. 24 bit color), then patterns of adjacent pixels are used to simulate the appearance of the unrepresented colors.

Edges are places in the image with strong intensity contrast. Since edges often occur at image locations representing object boundaries, edge detection is extensively used in image segmentation when we want to divide the image into areas corresponding to different objects. Representing an image by its edges has the further advantage that the amount of data is reduced significantly while retaining most of the image information.

Since edges consist of mainly high frequencies, we can, in theory, detect edges by applying a high pass frequency filter in the Fourier domain or by convolving the image with an appropriate kernel in the spatial domain. In practice, edge detection is performed in the spatial domain, because it is computationally less expensive and often yields better results. Since edges correspond to strong illumination gradients, we can highlight them by calculating the derivatives of the image. This is illustrated for the one-dimensional .

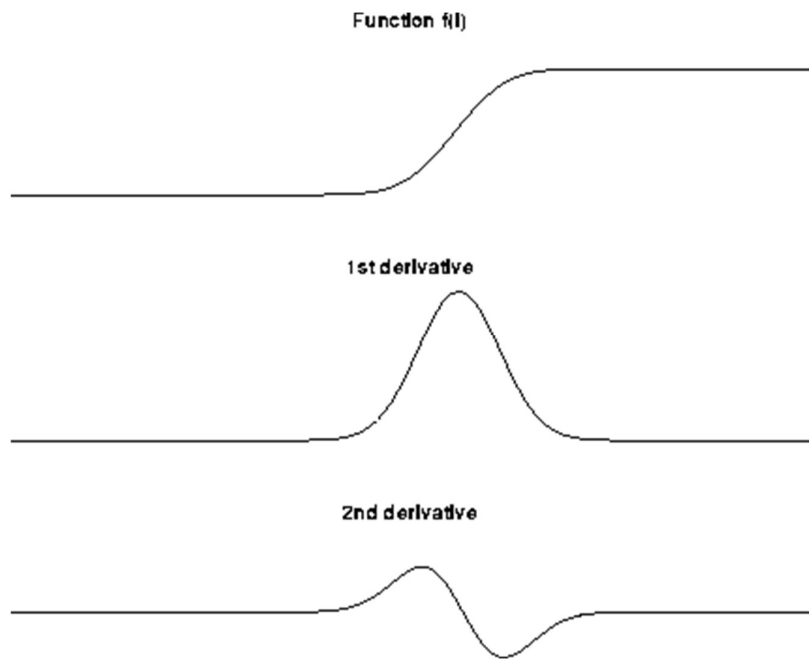


Fig 5.4 illustration graph 1st and 2nd derivative of an edge illustrated in one dimension.

We can see that the position of the edge can be estimated with the maximum of the 1st derivative or with the zero-crossing of the 2nd derivative. Therefore, we want to find a technique to calculate the derivative of a two-dimensional image. For a discrete one-dimensional function $f(i)$, the first derivative can be approximated by

$$f'(i) \approx \frac{f(i+1) - f(i-1)}{2}$$

Calculating this formula is equivalent to convolving the function with $[-1 \ 1]$. Similarly the 2nd derivative can be estimated by convolving $f(i)$ with $[1 \ -2 \ 1]$. Different edge detection kernels which are based on the above formula enable us to calculate either the 1st or the 2nd derivative of a two-dimensional image. There are two common approaches to estimate the 1st derivative in a two-dimensional image, Prewitt compass edge detection and gradient edge detection. Prewitt compass edge detection involves convolving the image with a set of (usually 8) kernels, each of which is sensitive to a different edge orientation. The kernel producing the maximum response at a pixel location determines the edge magnitude and orientation. Different sets of kernels might be used: examples include Prewitt, Sobel, Kirsch and Robinson kernels. Gradient edge detection is the second and more widely used technique. Here, the image is convolved with only two kernels, one estimating the gradient in the x-direction, G_x , the other the gradient in the y-direction, G_y .

In many implementations, the gradient magnitude is the only output of a gradient

edge detector, however the edge orientation might be calculated with the most common kernels used for the gradient edge detector are the Sobel, Roberts Cross and Prewitt operators. After having calculated the magnitude of the 1st derivative, we now have to identify those pixels corresponding to an edge. The easiest way is to threshold the gradient image, assuming that all pixels having a local gradient above the threshold must represent an edge. An alternative technique is to look for local maxima in the gradient image, thus producing one pixel wide edges. A more sophisticated technique is used by the Canny edge detector. It first applies a gradient edge detector to the image and then finds the edge pixels using non-maximal suppression and hysteresis tracking.

An operator based on the 2nd derivative of an image is the Marr edge detector, also known as zero crossing detector. Here, the 2nd derivative is calculated using a Laplacian of Gaussian (LoG) filter. The Laplacian has the advantage that it is an isotropic measure of the 2nd derivative of an image, i.e. the edge magnitude is obtained independently from the edge orientation by convolving the image with only one kernel. The edge positions are then given by the zero-crossings in the LoG image. The scale of the edges which are to be detected can be controlled by changing the variance of the Gaussian.

A general problem for edge detection is its sensitivity to noise, the reason being that calculating the derivative in the spatial domain corresponds to accentuating high frequencies and hence magnifying noise. This problem is addressed in the Canny and Marr operators by convolving the image with a smoothing operator (Gaussian) before calculating the derivative.

The frequency domain is a space in which each image value at image position F represents the amount that the intensity values in image I vary over a specific distance related to F . In the frequency domain, changes in image position correspond to changes in the spatial frequency, (or the rate at which image intensity values) are changing in the spatial domain image I .

For example, suppose that there is the value 20 at the point that represents the frequency 0.1 (or 1 period every 10 pixels). This means that in the corresponding spatial domain image I the intensity values vary from dark to light and back to dark

over a distance of 10 pixels, and that the contrast between the lightest and darkest is 40 gray levels (2 times 20).

The spatial frequency domain is interesting because: 1) it may make explicit periodic relationships in the spatial domain, and 2) some image processing operators are more efficient or indeed only practical when applied in the frequency domain. In most cases, the Fourier Transform is used to convert images from the spatial domain into the frequency domain and vice-versa.

A related term used in this context is spatial frequency, which refers to the (inverse of the) periodicity with which the image intensity values change. Image features with high spatial frequency (such as edges) are those that change greatly in intensity over short image distances.

A grayscale (or graylevel) image is simply one in which the only colors are shades of gray. The reason for differentiating such images from any other sort of color image is that less information needs to be provided for each pixel. In fact a 'gray' color is one in which the red, green and blue components all have equal intensity in RGB space, and so it is only necessary to specify a single intensity value for each pixel, as opposed to the three intensities needed to specify each pixel in a full color image.

Often, the grayscale intensity is stored as an 8-bit integer giving 256 possible different shades of gray from black to white. If the levels are evenly spaced then the difference between successive graylevels is significantly better than the graylevel resolving power of the human eye. Grayscale images are very common, in part because much of today's display and image capture hardware can only support 8-bit images. In addition, grayscale images are entirely sufficient for many tasks and so there is no need to use more complicated and harder-to-process color images.

There is a huge variety of software for manipulating images in various ways. Much of this software can be grouped under the heading image processing software, and the bulk of this reference is concerned with that group. Another very important category is what we call image editing software. This group includes painting programs, graphic art packages and so on. They are often useful in conjunction with image processing software packages, in situations where direct immediate

interaction with an image is the easiest way of achieving something. For instance, if a region of an image is to be masked out for subsequent image processing, it may be easiest to create the mask using an art package by directly drawing on top of the original image. The mask used in the description of the AND operator was created this way for instance. Art packages also often allow the user to move sections of the images around and brighten or darken selected regions interactively. Few dedicated image processing packages offer the same flexibility and ease of use in this respect.

Idempotence are Some operators have the special property that applying them more than once to the same image produces no further change after the first application. Such operators are said to be idempotent. Examples include the morphological operators opening and closing.

An isotropic operator in an image processing context is one which applies equally well in all directions in an image, with no particular sensitivity or bias towards one particular set of directions (e.g. compass directions). A typical example is the zero crossing edge detector which responds equally well to edges in any orientation. Another example is Gaussian smoothing. It should be borne in mind that although an operator might be isotropic in theory, the actual implementation of it for use on a discrete pixel grid may not be perfectly isotropic. An example of this is a Gaussian smoothing filter with very small standard deviation on a square grid.

A kernel is a (usually) smallish matrix of numbers that is used in image convolutions. Differently sized kernels containing different patterns of numbers give rise to different results under convolution. For instance, Figure 1 shows a 3×3 kernel that implements a mean filter.

1	1	1
1	1	1
1	1	1

Set of coordinate points =
 $\{ (-1, -1), (0, -1), (1, -1),$
 $(-1, 0), (0, 0), (1, 0),$
 $(-1, 1), (0, 1), (1, 1) \}$

Figure 5.5 Convolution kernel for a mean filter with 3×3 neighborhood.

The word 'kernel' is also commonly used as a synonym for 'structuring element', which is a similar object used in mathematical morphology. A structuring element differs from a kernel in that it also has a specified origin. This sense of the word 'kernel' is not used in HIPR. Logical operators are generally derived from Boolean algebra, which is a mathematical way of manipulating the truth values of concepts in an abstract way without bothering about what the concepts actually mean. The truth value of a concept in Boolean value can have just one of two possible values: true or false. Boolean algebra allows you to represent things like:

The block is both red and large by something like: $A \text{ AND } B$ where A represents 'The block is red', and B represents 'The block is large'. Now each of these sub-phrases has its own truth value in any given situation: each sub-phrase is either true or false. Moreover, the entire composite phrase also has a truth value: it is true if both of the sub-phrases are true, and false in any other case. We can write this AND combination rule (and its dual operation NAND) using a truth-table as shown in Figure 1, in which we conventionally represent true by 1, and

A	B	Q	A	B	Q
0	0	0	0	0	1
0	1	0	0	1	1
1	0	0	1	0	1
1	1	1	1	1	0

false by zero. **AND** **NAND**

Figure 5.6 Truth-tables for AND and NAND

The left hand table shows each of the possible combinations of truth values of A and B, and the the resulting truth value of $A \text{ AND } B$. Similar truth-tables can be set up for the other logical operators: NAND, OR, NOR, XOR, XNOR and NOT.

Turning now to an image processing context, the pixel values in a binary image, which are either 0 or 1, can be interpreted as truth values as above. Using this convention we can carry out logical operations on images simply by applying the truth-table combination rules to the pixel values from a pair of input images (or a single input image in the case of NOT). Normally, corresponding pixels from each of two identically sized binary input images are compared to produce the output image,

which is another binary image of the same size. As with other image arithmetic operations, it is also possible to logically combine a single input image with a constant logical value, in which case each pixel in the input image is compared to the same constant in order to produce the corresponding output pixel. See the individual logical operator descriptions for examples of these operations.

Logical operations can also be carried out on images with integer pixel values. In this extension the logical operations are normally carried out in bitwise fashion on binary representations of those integers, comparing corresponding bits with corresponding bits to produce the output pixel value. For instance, suppose that we wish to XOR the integers 47 and 255 together using 8-bit integers. 47 is 00101111 in binary and 255 is 11111111. XORing these together in bitwise fashion, we have 11010000 in binary or 208 in decimal.

Note that not all implementations of logical operators work in such bitwise fashion. For instance some will treat zero as false and any non-zero value as true and will then apply the conventional 1-bit logical functions to derive the output image. The output may be a simple binary image itself, or it may be a graylevel image formed perhaps by multiplying what would be the binary output image (containing 0's and 1's) with one of the input images.

Look-Up Tables or LUTs are fundamental to many aspects of image processing. An LUT is simply a table of cross-references linking index numbers to output values. The most common use is to determine the colors and intensity values with which a particular image will be displayed, and in this context the LUT is often called simply a colormap.

The idea behind the colormap is that instead of storing a definite color for each pixel in an image, for instance in 24-bit RGB format, each pixel's value is instead treated as an index number into the colormap. When the image is to be displayed or otherwise processed, the colormap is used to look up the actual colors corresponding to each index number. Typically, the output values stored in the LUT would be RGB color values.

There are two main advantages to doing things this way. Firstly, the index number

can be made to use fewer bits than the output value in order to save storage space. For instance, an 8-bit index number can be used to look up a 24-bit RGB color value in the LUT. Since only the 8-bit index number needs to be stored for each pixel, such 8-bit color images take up less space than a full 24-bit image of the same size. Of course, the image can only contain 256 different colors (the number of entries in an 8-bit LUT), but this is sufficient for many applications and usually the observable image degradation is small.

Secondly the use of a color table allows the user to experiment easily with different color labeling schemes for an image. One disadvantage of using a colormap is that it introduces additional complexity into an image format. It is usually necessary for each image to carry around its own colormap, and this LUT must be continually consulted whenever the image is displayed or processed.

Another problem is that in order to convert from a full color image to (say) an 8-bit color image using a color image, it is usually necessary to throw away many of the original colors, a process known as color quantization. This process is lossy, and hence the image quality is degraded during the quantization process. Additionally, when performing further image processing on such images, it is frequently necessary to generate a new colormap for the new images, which involves further color quantization, and hence further image degradation. As well as their use in colormaps, LUTs are often used to remap the pixel values within an image. This is the basis of many common image processing point operations such as thresholding, gamma correction and contrast stretching. The process is often referred to as anamorphosis.

A mask is a binary image consisting of zero- and non-zero values. If a mask is applied to another binary or to a grayscale image of the same size, all pixels which are zero in the mask are set to zero in the output image. All others remain unchanged. Masking can be implemented either using pixel multiplication or logical AND, the latter in general being faster. Masking is often used to restrict a point or arithmetic operator to an area defined by the mask. We can, for example, accomplish this by first masking the desired area in the input image and processing it with the operator, then masking the original input image with the inverted mask to obtain the unprocessed area of the image and finally recombining the two partial

images using image addition. An example can be seen in the worksheet on the logical AND operator. In some image processing packages, a mask can directly be defined as an optional input to a point operator, so that automatically the operator is only applied to the pixels defined by the mask .

The field of mathematical morphology contributes a wide range of operators to image processing, all based around a few simple mathematical concepts from set theory. The operators are particularly useful for the analysis of binary images and common usages include edge detection, noise removal, image enhancement and image segmentation. The two most basic operations in mathematical morphology are erosion and dilation. Both of these operators take two pieces of data as input: an image to be eroded or dilated, and a structuring element (also known as a kernel). The two pieces of input data are each treated as representing sets of coordinates in a way that is slightly different for binary and grayscale images.

For a binary image, white pixels are normally taken to represent foreground regions, while black pixels denote background. (Note that in some implementations this convention is reversed, and so it is very important to set up input images with the correct polarity for the implementation being used). Then the set of coordinates corresponding to that image is simply the set of two-dimensional Euclidean coordinates of all the foreground pixels in the image, with an origin normally taken in one of the corners so that all coordinates have positive elements.

For a grayscale image, the intensity value is taken to represent height above a base plane, so that the grayscale image represents a surface in three-dimensional Euclidean space. Figure 1 shows such a surface. Then the set of coordinates associated with this image surface is simply the set of three-dimensional Euclidean coordinates of all the points within this surface and also all points below the surface, down to the base plane. Note that even when we are only considering points with integer coordinates, this is a lot of points, so usually algorithms are employed that do not need to consider all the points.

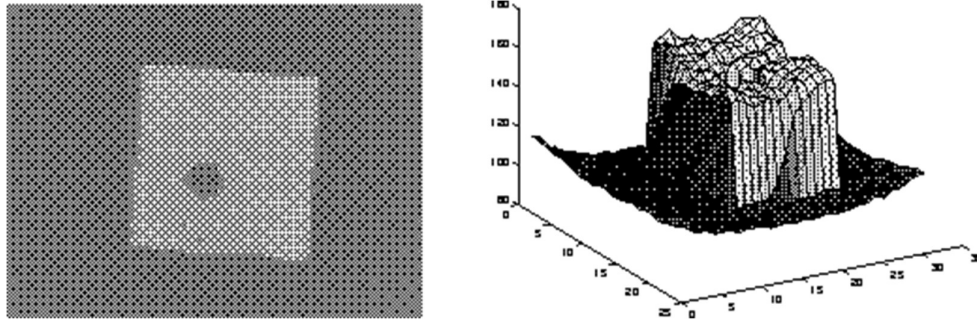


Figure 5.7 Simple graylevel image and the corresponding surface in image space

The structuring element is already just a set of point coordinates (although it is often represented as a binary image). It differs from the input image coordinate set in that it is normally much smaller, and its coordinate origin is often not in a corner, so that some coordinate elements will have negative values. Note that in many implementations of morphological operators, the structuring element is assumed to be a particular shape (e.g. a 3×3 square) and so is hardwired into the algorithm. Binary morphology can be seen as a special case of graylevel morphology in which the input image has only two graylevels at values 0 and 1.

Erosion and dilation work (at least conceptually) by translating the structuring element to various points in the input image, and examining the intersection between the translated kernel coordinates and the input image coordinates. For instance, in the case of erosion, the output coordinate set consists of just those points to which the origin of the structuring element can be translated, while the element still remains entirely 'within' the input image. Virtually all other mathematical morphology operators can be defined in terms of combinations of erosion and dilation along with set operators such as intersection and union. Some of the more important are opening, closing and skeletonization.

A multi-spectral image is a collection of several monochrome images of the same scene, each of them taken with a different sensor. Each image is referred to as a band. A well-known multi-spectral (or multi-band image) is a RGB color image, consisting of a red, a green and a blue image, each of them taken with a sensor sensitive to a different wavelength. In image processing, multi-spectral images are most commonly used for Remote Sensing applications. Satellites usually take several images from frequency bands in the visual and non-visual range. Landsat 5, for example, produces 7 band images with the wavelength of the bands being

between 450 and 1250 nm.

All the standard single-band image processing operators can also be applied to multi-spectral images by processing each band separately. For example, a multi-spectral image can be edge detected by finding the edges in each band and then ORing the three edge images together. However, we would obtain more reliable edges, if we associate a pixel with an edge based on its properties in all three bands and not only in one.

To fully exploit the additional information which is contained in the multiple bands, we should consider the images as one multi-spectral image rather than as a set of monochrome graylevel images. For an image with k bands, we can then describe the brightness of each pixel as a point in a k -dimensional space represented by a vector of length k .

Special techniques exist to process multi-spectral images. For example, to classify a pixel as belonging to one particular region, its intensities in the different bands are said to form a feature vector describing its location in the k -dimensional feature space. The simplest way to define a class is to choose a upper and lower threshold for each band, thus producing a k -dimensional 'hyper-cube' in the feature space. Only if the feature vector of a pixel points to a location within this cube, is the pixel classified as belonging to this class. A more sophisticated classification method is described in the corresponding worksheet.

The disadvantage of multi-spectral images is that, since we have to process additional data, the required computation time and memory increase significantly. However, since the speed of the hardware will increase and the costs for memory will decrease in the future, it can be expected that multi-spectral images will become more important in many fields of computer vision.

Suppose that an image processing operator F acting on two input images A and B produces output images C and D respectively. If the operator F is linear, then

$$F(a \times A + b \times B) = a \times C + b \times D \quad \text{--eq5}$$

where a and b are constants. In practice, this means that each pixel in the output of a linear operator is the weighted sum of a set of pixels in the input image. By contrast, non-linear operators are all the other operators. For example, the threshold operator is non-linear, because individually, corresponding pixels in the two images A and B may be below the threshold, whereas the pixel obtained by adding A and B may be above threshold. Similarly, an absolute value operation is non-linear:

$$|-1 + 1| \neq |-1| + |1| \text{--eq6}$$

as is the exponential operator:

$$\exp(1 + 1) \neq \exp(1) + \exp(1) \text{--eq7}$$

In order for any digital computer processing to be carried out on an image, it must first be stored within the computer in a suitable form that can be manipulated by a computer program. The most practical way of doing this is to divide the image up into a collection of discrete (and usually small) cells, which are known as pixels. Most commonly, the image is divided up into a rectangular grid of pixels, so that each pixel is itself a small rectangle. Once this has been done, each pixel is given a pixel value that represents the color of that pixel. It is assumed that the whole pixel is the same color, and so any color variation that did exist within the area of the pixel before the image was discretized is lost. However, if the area of each pixel is very small, then the discrete nature of the image is often not visible to the human eye.

Other pixel shapes and formations can be used, most notably the hexagonal grid, in which each pixel is a small hexagon. This has some advantages in image processing, including the fact that connectivity is less ambiguously defined than with a square grid, but hexagonal grids are not widely used. Part of the reason is that many image capture systems (e.g. most CCD cameras and scanners) intrinsically discretize the captured image into a rectangular grid in the first instance.

The notation of pixel connectivity describes a relation between two or more pixels. For two pixels to be connected they have to fulfill certain conditions on the pixel brightness and spatial adjacency. First, in order for two pixels to be considered connected, their pixel values must both be from the same set of values V. For a grayscale image, V might be any range of graylevels, e.g. $V=\{22,23,\dots,40\}$, for a

binary image we simple have $V=\{1\}$.To formulate the adjacency criterion for connectivity, we first introduce the notation of neighborhood. For a pixel p with the coordinates (x,y) the set of pixels given by:

$$N_4(p) = \{(x + 1, y), (x - 1, y), (x, y + 1), (x, y - 1)\} \text{---eq8}$$

is called its 4-neighbors. Its 8-neighbors are defined as

$$N_8(p) = N_4 \cup \{(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)\} \text{---eq9}$$

From this we can infer the definition for 4- and 8-connectivity:

Two pixels p and q , both having values from a set V are 4-connected if q is from the set $N_4(p)$ and 8-connected if q is from $N_8(p)$.

General connectivity can either be based on 4- or 8-connectivity; for the following discussion we use 4-connectivity. A pixel p is connected to a pixel q if p is 4-connected to q or if p is 4-connected to a third pixel which itself is connected to q . Or, in other words, two pixels q and p are connected if there is a path from p and q on which each pixel is 4-connected to the next one. A set of pixels in an image which are all connected to each other is called a connected component. Finding all connected components in an image and marking each of them with a distinctive label is called connected component labeling.

An example of a binary image with two connected components which are based on 4-connectivity can be seen in Figure 1. If the connectivity were based on 8-neighbors, the two connected components would merge into one

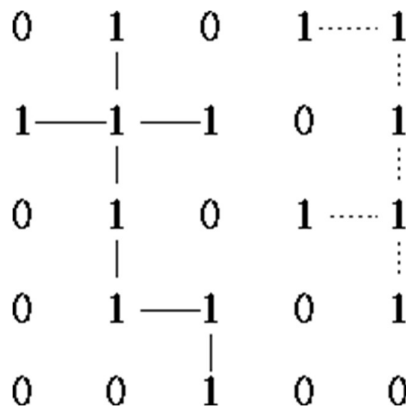


Figure 5.8 Two connected components based on 4-connectivity.

Pixel Values are the Each of the pixels that represents an image stored inside a computer has a pixel value which describes how bright that pixel is, and/or what color it should be. In the simplest case of binary images, the pixel value is a 1-bit number indicating either foreground or background. For a grayscale images, the pixel value is a single number that represents the brightness of the pixel. The most common pixel format is the byte image, where this number is stored as an 8-bit integer giving a range of possible values from 0 to 255. Typically, zero is taken to be black, and 255 is taken to be white. Values in between make up the different shades of gray.

To represent color images, separate red, green and blue components must be specified for each pixel (assuming an RGB colorspace), and so the pixel 'value' is actually a vector of three numbers. Often the three different components are stored as three separate 'grayscale' images known as color planes (one for each of red, green and blue), which have to be recombined when displaying or processing.

Multi-spectral images can contain even more than three components for each pixel, and by extension these are stored in the same kind of way, as a vector pixel value, or as separate color planes. The actual grayscale or color component intensities for each pixel may not actually be stored explicitly. Often, all that is stored for each pixel is an index into a colormap in which the actual intensity or colors can be looked up.

Although simple 8-bit integers or vectors of 8-bit integers are the most common sorts of pixel values used, some image formats support different types of value, for instance 32-bit signed integers or floating point values. Such values are extremely useful in image processing as they allow processing to be carried out on the image where the resulting pixel values are not necessarily 8-bit integers. If this approach is used then it is usually necessary to set up a colormap which relates particular ranges of pixel values to particular displayed colors.

It is a useful fact that the huge variety of colors that can be perceived by humans can all be produced simply by adding together appropriate amounts of red, blue and green colors. These colors are known as the primary colors. Thus in most image processing applications, colors are represented by specifying separate intensity values for red, green and blue components. This representation is commonly

referred to as RGB. The primary color phenomenon results from the fact that humans have three different sorts of color receptors in their retinas which are each most sensitive to different visible light wavelengths. The primary colors used in painting (red, yellow and blue) are different. When paints are mixed, the 'addition' of a new color paint actually subtracts wavelengths from the reflected visible light.

A color perceived by the human eye can be defined by a linear combination of the three primary colors red, green and blue. These three colors form the basis for the RGB-colorspace. Hence, each perceivable color can be defined by a vector in the three-dimensional colorspace. The intensity is given by the length of the vector, and the actual color by the two angles describing the orientation of the vector in the colorspace. The RGB-space can also be transformed into other coordinate systems, which might be more useful for some applications. One common basis for the color space is IHS. In this coordinate system, a color is described by its intensity, hue (average wavelength) and saturation (the amount of white in the color). This color space makes it easier to directly derive the intensity and color of perceived light and is therefore more likely to be used by human beings.

For simplicity, assume that the image I being considered is formed by projection from scene S (which might be a two- or three-dimensional scene, etc.). The spatial domain is the normal image space, in which a change in position in I directly projects to a change in position in S . Distances in I (in pixels) correspond to real distances (e.g. in meters) in S .

This concept is used most often when discussing the frequency with which image values change, that is, over how many pixels does a cycle of periodically repeating intensity variations occur. One would refer to the number of pixels over which a pattern repeats (its periodicity) in the spatial domain. In most cases, the Fourier Transform will be used to convert images from the spatial domain into the frequency domain.

A related term used in this context is spatial frequency, which refers to the (inverse of the) periodicity with which the image intensity values change. Image features with high spatial frequency (such as edges) are those that change greatly in intensity over short image distances. Another term used in this context is spatial derivative,

which refers to how much the image intensity values change per change in image position.

The field of mathematical morphology provides a number of important image processing operations, including erosion, dilation, opening and closing. All these morphological operators take two pieces of data as input. One is the input image, which may be either binary or grayscale for most of the operators. The other is the structuring element. It is this that determines the precise details of the effect of the operator on the image. The structuring element is sometimes called the kernel, but we reserve that term for the similar objects used in convolutions.

The structuring element consists of a pattern specified as the coordinates of a number of discrete points relative to some origin. Normally cartesian coordinates are used and so a convenient way of representing the element is as a small image on a rectangular grid. Figure 1 shows a number of different structuring elements of various sizes. In each case the origin is marked by a ring around that point. The origin does not have to be in the center of the structuring element, but often it is. As suggested by the figure, structuring elements that fit into a 3×3 grid with its origin at the center are the most commonly seen type.

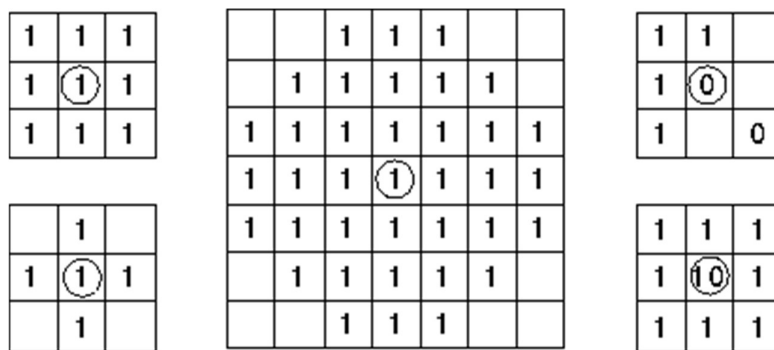


Figure 5.9 Some example structuring elements.

Note that each point in the structuring element may have a value. In the simplest structuring elements used with binary images for operations such as erosion, the elements only have one value, conveniently represented as a one. More complicated elements, such as those used with thinning or grayscale morphological operations, may have other pixel values. An important point to note is that although a rectangular grid is used to represent the structuring element, not every point in

that grid is part of the structuring element in general. Hence the elements shown in Figure 1 contain some blanks. In many texts, these blanks are represented as zeros, but this can be confusing and so we avoid it here. When a morphological operation is carried out, the origin of the structuring element is typically translated to each pixel position in the image in turn, and then the points within the translated structuring element are compared with the underlying image pixel values. The details of this comparison, and the effect of the outcome depend on which morphological operator is being used.

If an image is represented in a byte or integer pixel format, the maximum pixel value is limited by the number of bits used for the representation, e.g. the pixel values of a 8-bit image are limited to 255. However, many image processing operations produce output values which are likely to exceed the given maximum value. In such cases, we have to decide how to handle this pixel overflow. A pixel is generally thought of as the smallest single component of a digital image. However, the definition is highly context-sensitive. For example, there can be "printed pixels" in a page, or pixels carried by electronic signals, or represented by digital values, or pixels on a display device, or pixels in a digital camera (photosensor elements). This list is not exhaustive and, depending on context, synonyms include pel, sample, byte, bit, dot, and spot. *Pixels* can be used as a unit of measure such as: 2400 pixels per inch, 640 pixels per line, or spaced 10 pixels apart.

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 SYSTEM ARCHITECTURE

Describing the overall features of the software is concerned with defining the requirements and establishing the high level of the system. During architectural design, the various web pages and their interconnections are identified and designed. The major software components are identified and decomposed into processing modules and conceptual data structures and the interconnections among the modules are identified. The following modules are identified in the proposed system.

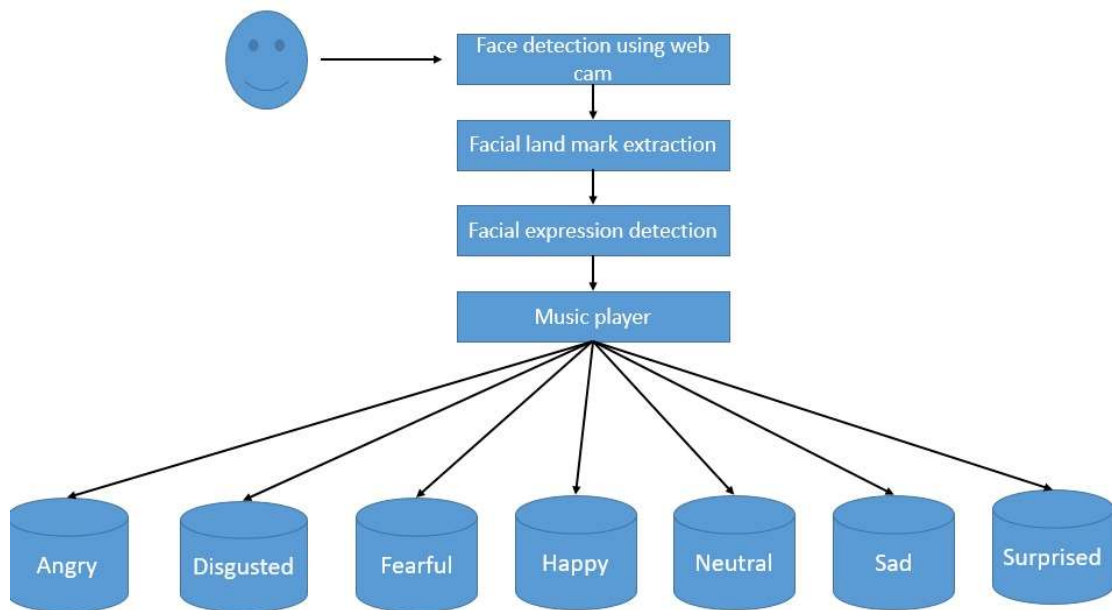


Fig 6.1 system architecture

CHAPTER 7

SYSTEM TESTING

7.1 Test plan

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also to assess the feature of a software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words software testing is a verification and validation process

7.2 Verification

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

7.3 Validation

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

7.4 Basics of software testing

There are two basics of software testing: black box testing and white box testing.

7.5 Black box Testing

Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.

7.6 White box Testing

White box testing is a testing technique that takes into account the internal

mechanism of a system. It is also called structural testing and glass box testing. Black box testing is often used for validation and white box testing is often used for verification.

7.7 Types of testing

There are many types of testing like

- Unit Testing
- Integration Testing
- Functional Testing
- System Testing
- Stress Testing
- Performance Testing
- Usability Testing
- Acceptance Testing
- Regression Testing
- Beta Testing

7.7.1 Unit Testing

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

7.7.2 Integration Testing

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

7.7.3 Functional Testing

Functional testing is the testing to ensure that the specified functionality required in the system requirements works. It falls under the class of black box testing.

7.7.4 System Testing

System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing.

7.7.5 Stress Testing

Stress testing is the testing to evaluate how system behaves under unfavorable conditions. Testing is conducted at beyond limits of the specifications. It falls under the class of black box testing.

7.7.6 Performance Testing

Performance testing is the testing to assess the speed and effectiveness of the system and to make sure it is generating results within a specified time as in performance requirements. It falls under the class of black box testing.

7.7.7 Usability Testing

Usability testing is performed to the perspective of the client, to evaluate how the GUI is user-friendly? How easily can the client learn? After learning how to use, how proficiently can the client perform? How pleasing is it to use its design? This falls under the class of black box testing.

7.7.8 Acceptance Testing

Acceptance testing is often done by the customer to ensure that the delivered product meets the requirements and works as the customer expected. It falls under the class of black box testing.

7.7.9 Regression Testing

Regression testing is the testing after modification of a system, component, or a group of related units to ensure that the modification is working correctly and is not damaging or imposing other modules to produce unexpected results. It falls under

the class of black box testing

Requirement analysis, also called requirement engineering, is the process of determining user expectations for a new modified product. It encompasses the tasks that determine the need for analyzing, documenting, validating and managing software or system requirements. The requirements should be documentable, actionable, measurable, testable and traceable related to identified business needs or opportunities and define to a level of detail, sufficient for system design.

Functional requirement is a technical specification requirement for the software products. It is the first step in the requirement analysis process which lists the requirements of particular software systems including functional, performance and security requirements. The function of the system depends mainly on the quality hardware used to run the software with given functionality.

Usability specifies how easy the system must be use. It is easy to ask queries in any format which is short or long, porter stemming algorithm stimulates the desired response for user. Robustness refers to a program that performs well not only under ordinary conditions but also under unusual conditions. It is the ability of the user to cope with errors for irrelevant queries during execution.

Security states of providing protected access to resource is security. The system provides good security and unauthorized users cannot access the system there by providing high security.

Reliability is the probability of how often the software fails. The measurement is often expressed in MTBF (Mean Time Between Failures). The requirement is needed in order to ensure that the processes work correctly and completely without being aborted. It can handle any load and survive and survive and even capable of working around any failure.

Compatibility is supported by version above all web browsers. Using any web servers like localhost makes the system real-time experience. The flexibility of the project is provided in such a way that is has the ability to run on different environments being executed by different users. Safety is a measure taken to prevent trouble. Every query is processed in a secured manner without letting others to know one's personal information. Portability is the usability of the same software in different environments. The project can be run in any operating system

CHAPTER 8

RESULTS AND DISCUSSION

This proposed study proposes a Human face emotion and song playing either according to the current mood of the user. Human face images are captured with the help of a camera. Once the picture has been taken, the captured frame of the image is then converted into a gray scale image to improve the performance of the classifier which will be followed by sending the converted image to the classifier algorithm where individual features are extracted from the face. Several performing the function will help the classifier to get trained even better. The respective emotion gets detected and recognized with the help of patterns and then making wise decisions will help the user to get his mood better.

8.1 Facial emotions results:

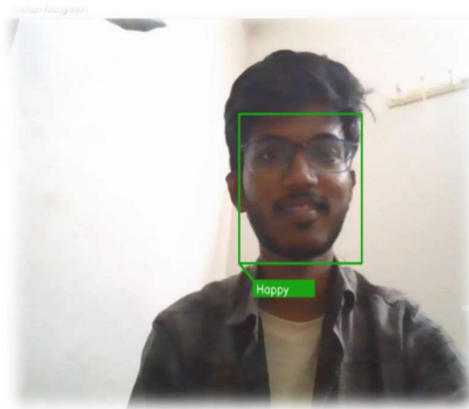


Fig8.1.1Happy

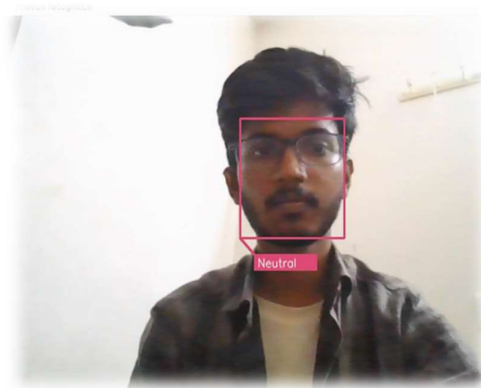


fig8.1.2Neutral

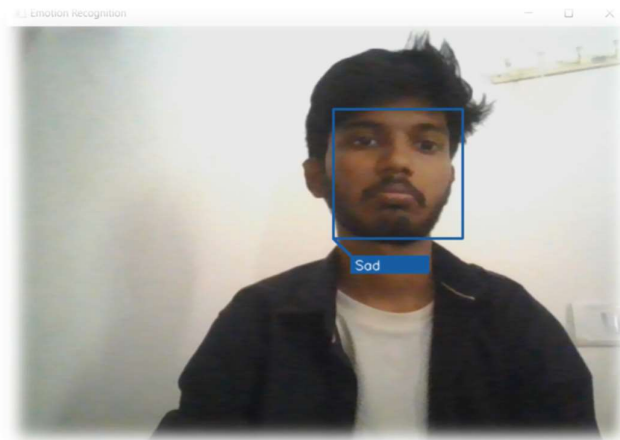


Fig8.1.3Sad

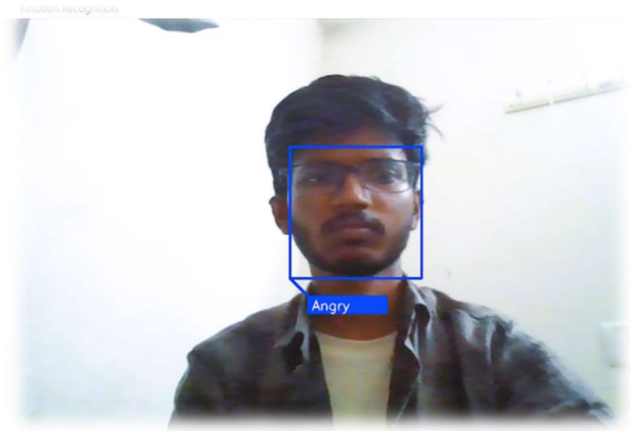


Fig8.1.4 Angry

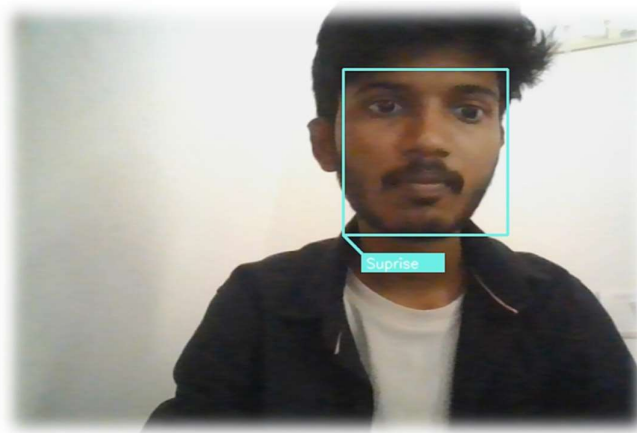


Fig 8.1.5 Surprise

8.2 Vocal based emotions results:



Fig8.2.1 Import Data



Fig8.2.2 Pre-processed Data



Fig8.2.3 Trained Data

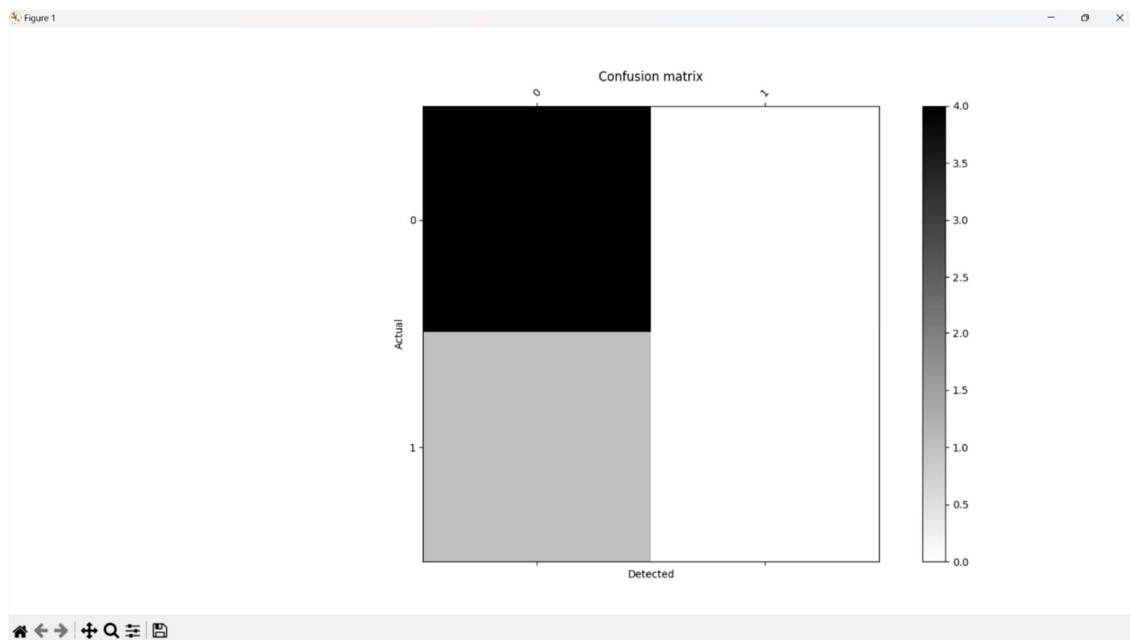


Fig 8.2.4 Graph 1

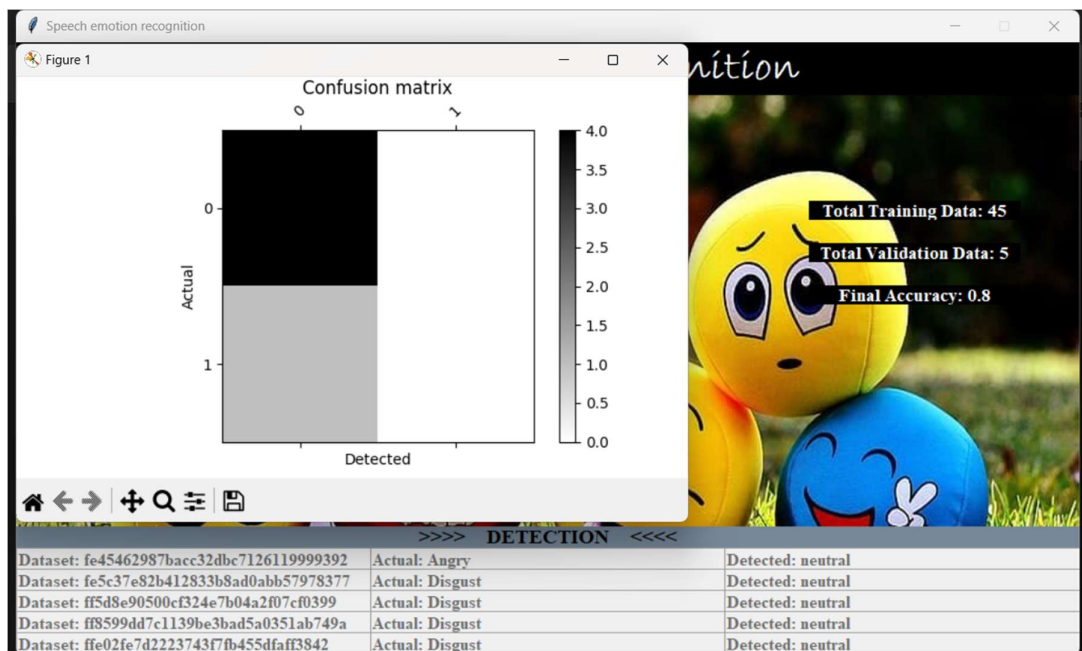


Fig 8.2.5 Graph2

CHAPTER 9

CONCLUSION

In this project, two CNN models were created to remove facial elements and perceive feelings. 7,074 images representing five emotions were selected from the FER 2013 dataset. The feelings that were taken into account were joy, sadness, rage, fear, and neutral. Landmark features were identified and extracted from these images after they were converted into NumPy arrays. The four phases of a CNN model were convolution, pooling, batch normalization, and dropout layers in the first three phases. The flatten, dense, and output layers make up the final phase. With fewer layers, a new CNN model was developed. There are 3,510,469 parameters in CNN Model 1, of which 3,507,781 can be trained, while Model 2 has 2,734,085 parameters, of which 2,732,293 can be trained. The accuracy and loss metrics were used to determine the ideal parameter values for these models. Five feelings were assessed utilizing accuracy, review and F score. Four and five different emotions were used to evaluate the models. With four emotions, CNN Model 1 had an average accuracy of 80.0% and a loss of 0.63, while with five emotions, it had an average accuracy of 72.2% and a loss of 0.85. With four emotions, CNN Model 2 had an average accuracy of 79.5 percent and a loss of 0.59, while with five emotions, it had an accuracy of 72.5 percent and a loss of 0.84. For CNN Model 1, the happy emotion had the highest precision, recall, and F-score of 0.88, while for CNN Model 2, it had the lowest precision, recall, and F-score of 0.87. With precision of 0.56, recall of 0.64, and F-score of 0.60 for CNN Model 1, and precision of 0.56, recall of 0.68, and F-score of 0.61 for Model 2, the neutral emotion performed the worst. The proposed models performed better when compared to other approaches described in the literature.

REFERENCES

- [1] Emanuel I. Andelin and Alina S. Rusu, "Investigation of facial microexpressions of emotions in psychopathy - a case study of an individual in detention", 2015, Published by Elsevier Ltd.
- [2] Paul Ekman, Wallace V Friesen, and Phoebe Ellsworth. Emotion in the human face: Guidelines for research and an integration of findings. Elsevier 2013.
- [3] F. De la Torre and J. F. Cohn, "Facial expression analysis," Vis. Anal. Hum., pp. 377–410, 2011.
- [4] Bavkar, Sandeep, Rangole, Jyoti, Deshmukh, " Geometric Approach for Human Emotion Recognition using Facial Expression", International Journal of Computer Applications, 2015.
- [5] Zhang, Z. Feature-based facial expression recognition: Sensitivity analysis and experiments with a multilayer perceptron. International Journal of Pattern Recognition and Artificial Intelligence.
- [6] Remi Delbouys, Romain Hennequin, Francesco Piccoli, Jimena RoyoLetelier, Manuel Moussallam. "Music mood detection based on audio.
- [7] nd lyrics with Deep Neural Net", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.
- [8] Krittrin Chankuptarat, et al, "Emotion Based Music Player", IEEE 2019 conference.
- [9] Kim, Y.: Convolutional Neural Networks for Sentence Classification. In: Proceedings of the 2014 Conference on EMNLP, pp. 1746–1751 (2014).
- [10] Tripathi, S., Beigi, H.: Multi-Modal Emotion recognition on IEMOCAP Dataset using Deep Learning. In: arXiv:1804.05788 (2018).
- [11] Teng et al., "Recognition of Emotion with SVMs", Lecture Notes in Computer Science, August 2006.
- [12] B.T. Nguyen, M.H. Trinh, T.V. Phan, H.D. Nguyen An efficient realtime emotion detection using camera and facial landmarks , 2017 seventh international conference on information science and technology (ICIST) (2017).
- [13] H. Immanuel James, J. James Anto Arnold, J. Maria Masilla Ruban, M. Tamilarasan, R. Saranya" EMOTION BASED MUSIC RECOMMENDATION SYSTEM": pISSN: 2395-0072 , IRJET 2019.

- [14] Hafeez Kabani, Sharik Khan , Omar Khan , Shabana Tadvil "Emotion Based Music Player" International Journal of Engineering Research and General Science Volume 3, Issue 1, January-February , 2015.
- [15] Shlok Gikla, Husain Zafar, Chuntan Soni, Kshitija Waghurdekar "SMART MUSIC INTEGRATING AND MUSIC MOOD RECOMMENDATION" 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSpNet).
- [16] Srushti Sawant, Shraddha Patil, Shubhangi Biradar, "EMOTION BASED MUSIC SYSTEM", International Journal of Innovations & Advancement in Computer Science, IJIACS ISSN 2347-8616 volume 7, Issue 3 March 2018.
- [17] Mutasem K. Alsmadi "FACIAL EXPRESSION RECOGNITION" US2018/0211102A1.
- [18] Raghuvanshi, Arushi, and Vivek Choksi. "Facial Expression Recognition with Convolutional Neural Network" Stanford University, 2018.
- [19] McFee, Brian, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. "Libros: Audio and music signal analysis in python." In Proceedings of the 14th python in science conference, pp. 18-25. 2015. [9] Giannakopoulos, Theodoros."
- [20] Xiao Luan, Huaming Wang, Weihua Ou and Linghui Liu "Facial expressions Detection using cnn" 2017 International Conference on SPAC Security, Pattern Analysis.
- [21] Bavkar, Sandeep, Rangole, Jyoti, Deshmukh, "Geometric Approach for Human Emotion Recognition using Facial Expression", International Journal of Computer Applications, 2015.
- [22] Krittrin Chankuptarat, et al, "Emotion Based Music Player", IEEE 2019 conference.
- [23] Tripathi, S., Beigi, H.: Multi-Modal Emotion recognition on IEMOCAP Dataset using Deep Learning. In: arXiv:1804.05788 (2018).
- [24] Teng et al., "Recognition of Emotion with SVMs", Lecture Notes in Computer Science, August 2006.
- [25] B.T. Nguyen, M.H. Trinh, T.V. Phan, H.D. Nguyen "An efficient realtime emotion detection using camera and facial landmarks, 2017 seventh international conference on information science and technology (ICIST) (2017).

- [26] K. Kaulard, D.W. Cunningham, H.H. Bulthoff, C. Wallraven, The MPI facial expression database: A validated database of emotional and conversational facial expressions, PLoS One, vol. 7, no. 3, art. e32321, (2021).
- [27] G.E. Hinton et al., Deep neural networks for acoustic modelling in speech recognition: The shared views of four research groups, IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 82-97, (2021).
- [28] A. Pentland, Social signal processing, IEEE Signal Processing Magazine, vol. 24, no. 4, pp. 108- 111, (2021).
- [29] M.Xie, Development of artificial intelligence and effects on financial system, Journal of Physics: Conference Series 1187, art. 032084, (2020).
- [30] A. Nandi, F.Xhafa, L.Subirats, S. Fort, Real time emotion classification using electroencephalogram data stream in e-learning contexts, Sensors, vol. 21, no. 5, art. 1589, (2021).
- [31] A. Raheel, M. Majid, S.M. Anwar, M. Alnowami, Physiological sensors-based emotion recognition while experiencing tactile enhanced multimedia, Sensors, vol. 20, no. 14, art. 04037, (2020).
- [32] D. Keltner, P. Ekman M. Lewis, J.H. Jones, Handbook of Emotions (2nd ed.), Guilford Publications (New York), pp. 236-249, (2020).