Texture based Feature Extraction Strategies for Facial Expression Recognition

Submitted in partial fulfillment of the requirements

for the award of the degree of

DOCTOR OF PHILOSOPHY

Submitted by

Mukku Nisanth Kartheek

(Roll No. 717147)

Under the guidance of

Dr. Munaga V. N. K. Prasad

and

Dr. Raju Bhukya

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL

TELANGANA - 506004, INDIA

October 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
TELANGANA - 506004, INDIA

THESIS APPROVAL FOR Ph.D.

This is to certify that the thesis entitled, Texture based Feature Extraction Strategies for Facial Expression Recognition, submitted by Mr. Mukku Nisanth Kartheek [Roll No. 717147] is approved for the degree of DOCTOR OF PHILOSOPHY at National Institute of Technology Warangal.

Examiner

Research Supervisor
Dr. Munaga V N K Prasad
Center for Affordable Technologies
Institute for Development &
Research in Banking Technology
India

Research Supervisor
Dr. Raju Bhukya
Dept. of Computer Science and Engg.
NIT Warangal
India

Chairman
Dr. S. Ravi Chandra
Head, Dept. of Computer Science and Engg.
NIT Warangal
India

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
TELANGANA - 506004, INDIA

CERTIFICATE

This is to certify that the thesis entitled, Texture based Feature Extraction Strategies for Facial Expression Recognition, submitted in partial fulfillment of requirement for the award of degree of DOCTOR OF PHILOSOPHY to National Institute of Technology Warangal, is a bonafide research work done by Mr. Mukku Nisanth Kartheek [Roll No. 717147] under our supervision. The contents of the thesis have not been submitted elsewhere for the award of any degree.

Research Supervisor
Dr. Munaga V N K Prasad
Center for Affordable Technologies
Institute for Development &
Research in Banking Technology
India

Hyderabad
Date: 31-10-2022

Research Supervisor
Dr. Raju Bhukya
Dept. of Computer Science and Engg.
NIT Warangal
India

Warangal
Date: 31-10-2022

DECLARATION

This is to certify that the work presented in the thesis entitled "*Texture based Feature Extraction Strategies for Facial Expression Recognition*" is a bonafide work done by me under the supervision of Dr. Munaga V. N. K. Prasad and Dr. Raju Bhukya. The work was not submitted elsewhere for the award of any degree.

I, hereby, declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / date / fact / source in my submission. I understand that any violation of the above will be cause for disciplinary action by the institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Mukku Nisanth Kartheek

(Roll No. 717147)

Date: 31-10-2022

ACKNOWLEDGMENT

Every day spent during my Ph.D has provided me an opportunity to enhance my learning. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this thesis. First and foremost, I sincerely appreciate the almighty God, who has granted countless blessings, strength and knowledge throughout my life. His benevolence has made me excel and successful in all my academic pursuits.

I am extremely grateful to my supervisors, Dr. Munaga V. N. K. Prasad, Associate Professor, Institute for Development and Research in Banking Technology (IDRBT), Hyderabad and Dr. Raju Bhukya, Associate Professor, Department of Computer Science and Engineering, National Institute of Technology (NIT), Warangal for their valuable guidance, continuous support, and patience during my Ph.D study. Their immense knowledge and plentiful experience have helped me throughout my academic research.

I extend my gratitude to the Doctoral Scrutiny Committee (DSC) members comprising of Prof. D.V.L.N. Somayajulu, Dr. S. Ravi Chandra, Dr. U. S. N. Raju and Prof. Ravi Kumar Jatoth for their insightful comments and suggestions during my oral presentations. I am immensely thankful to Prof. Ch. Sudhakar, Prof. R. B. V. Subramaanyam, Prof. P. Radha Krishna and Dr. S. Ravi Chandra, Heads of Department of Computer Science and Engineering (CSE) and chairmans of DSC, during my tenure for providing adequate facilities in the Department to carry out the oral presentations.

I wish to express my sincere thanks to Prof. N.V. Ramana Rao (Director, NIT Warangal), Dr. A. S. Ramasastri and Prof. D. Janakiram (Director, IDRBT, Hyderabad) for providing the infrastructure and facilities to carry out my research. I am also very much grateful to the faculty members of CSE Department, NIT Warangal and IDRBT for their moral support throughout my research work.

On the personal level, I would also like to thank my scholar friends at IDRBT and NIT Warangal for their valuable suggestions and for extending selfless cooperation. Last but not the least, I would like to express my gratitude to my family. Without their tremendous understanding and encouragement in the past few years, it would be impossible for me to complete my study.

Mukku Nisanth Kartheek

ABSTRACT

Facial expressions form an important part of non-verbal communication as they provide an immediate means to analyze the mood of a person. In Automatic Facial Expression Recognition (AFER) systems, the main task is to extract significant and discriminative features from the facial images that can best classify the expressions into various categories. From one expression to another expression, there are only minute differences and the texture based methods have been found to effectively capture those minute changes such as skin changes, wrinkles, edges on the facial images. For accurately detecting facial expressions, the relationship of neighboring pixels and the relationship of adjacent pixels with the reference pixel is essential for detecting finer appearance changes with respect to various facial expressions. This motivated us towards proposing some local texture based feature descriptors for enhancing the recognition accuracy of AFER systems.

Chapter 1 in this thesis provides an introduction related to Facial Expression Recognition (FER), with an emphasis on feature extraction approaches. Next, the various benchmark datasets, experimental setup and performance measures related to FER have been presented. Finally towards the end, the research objectives, overall contributions and organization of the thesis have been mentioned. In Chapter 2, the related and the relevant state-of-the-art methods in the field of FER systems have been described. Towards the end of this Chapter, the research findings analyzed from the literature studies have been summarized.

Inspired by the Chess game rules, in Chapter 3, Chess Pattern (CP), Knight Tour Patterns (kTP and KTP) and in Chapter 4, Radial Mesh Pattern (RMP), Radial Cross Pattern (RCP), Chess Symmetric Pattern (CSP) and Radial Cross Symmetric Pattern (RCSP) feature descriptors have been proposed for extracting the facial features in a local neighborhood. CP has been proposed with an intention to generate different feature codes for corner, edge and flat portions of an image. Inspired by the Knight tour problem in graph theory, kTP (extracts features in a 3 x 3 neighborhood) and KTP (extracts features in a 5 x 5 neighborhood) feature descriptors have been proposed, which utilizes Knight moves for generating the features. RMP generates two feature codes that are unique to corner, edge

and flat portions of an image. RCP, CSP and RCSP feature descriptors have been proposed for overcoming some of the limitations of the existing methods such as CP, Local Gradient Coding (LGC) and it's variants.

In Chapter 5, feature descriptors inspired by the shape of various graphs such as Windmill graph, Generalised Petersen graph (GPG) and Triangle graph have been proposed for facial feature extraction. Windmill Graph based Feature Descriptor methods ($WGFD_h$ and $WGFD_v$) extract features by encoding the adjacent pixel relationship and the neighboring pixel relationship in a local neighborhood. The concept of using different weights have been applied to the proposed CP, kTP, KTP, RMP, RCP, CSP, RCSP, $WGFD_h$ and $WGFD_v$ methods for determining the optimal recognition accuracy. Petersen Graph based Binary Pattern (PGBP) extracts features based on the vertices and edges of GPG(6,2) in a local 5 x 5 neighborhood. Local Triangular Patterns (LTrP) extracts features by considering the triangles in both vertical and horizontal directions. The features are extracted in both clockwise and counter clockwise directions using the proposed PGBP and LTrP methods.

In Chapter 6, Feed Forward Neural Network Structure Inspired Feature Descriptors ($FFNND_1$ and $FFNND_2$) have been proposed to extract salient features in a local neighborhood. $FFNND_1$ method extracts three features by capturing the adjacent pixel relationship based on multi-distance information as like Local Mesh Pattern (LMeP), whereas, $FFNND_2$ method extracts two features by capturing the relationship between the pixels located at a radius (rd=2), from a reference pixel. For all the Chapters 3, 4, 5 and 6, the experiments have been performed with respect to both six and seven expressions on different 'in the lab' datasets in person independent setup to simulate a real world scenario. The experiments have also been performed with respect to seven expressions on Real World Affective Faces (RAF) and Facial Expression Research Group (FERG) datasets. In addition to six and seven expressions, the experiments have also been performed for eight expressions on Taiwanese Facial Expression Image Database (TFEID) and for ten expressions on Amsterdam Dynamic Facial Expression Set (ADFES) datasets respectively. The experimental results demonstrated the efficiency of the proposed methods when compared to the existing methods.

Contents

List of Figures

List of Tables

List of Algorithms

Glossary

AC-GAN Auxiliary Classifier Generative Adversarial Network

ACN Attentional Convolutional Network

ADFES Amsterdam Dynamic Facial Expression Set

AFER Automated Facial Expression Recognition

AFM Adaptive Feature Mapping

AI Artificial Intelligence

ALDP Angled Local Directional Pattern

ATM Automated Teller Machine

AU Action Units

BRS Biometric Recognition System

CK+ Extended Cohn-Kanade dataset

CMY Cyan, Magenta and Yellow

CNN Convolutional Neural Network

CP Chess Pattern

CRIP Cross-Centroid Ripple Pattern

CSLBP Center Symmetric Local Binary Pattern

CSLGC Center Symmetric Local Gradient Coding

CSLOP Center Symmetric Local Octonary Pattern

CSP Chess Symmetric Pattern

DA Data Augmentation

DAGSVM Directed Acyclic Graph Support Vector Machine

DAMCNN Deep Attentive Multi-Path Convolutional Neural Network

DBN Deep Belief Network

DCFA-CNN Deep Cross Feature Adaptive Network

DICNN Dual Integrated Convolutional Neural Network

DJSTN Deep Spatio Temporal Network

DL Deep Learning

DLFS Dictionary Learning Feature Space

DLPCNN Deep Locality Preserving Convolutional Neural Network

DSNGE Dual Non-negative Graph Embedding

DT Decision Tree

ED Euclidean Distance

FER Facial Expression Recognition

FER-net Facial Expression Recognition-Network

FERG Facial Expression Research Group

FFNND Feed Forward Neural Network Structure Inspired Feature Descriptors

FN False Negative

FP	False Positive
GAM	Geometric Appearance Models
GBPSC	Gammadion Binary Pattern of Shearlet Co-efficients
GPG	Generalized Petersen Graph
GSA	Gravitational Search Algorithm
HD	Hamming Distance
HCI	Human Computer Interaction
HiNet	Hybrid Inherited Feature Learning Network
HMM	Hidden Markov Model
HOG	Histogram of Oriented Gradients
ID	Identity
IDA	Information Discriminant Analysis
IFRBC	Individual Free Representation Based Classification
IFSL	Image Filter based Subspace Learning
JAFFE	Japanese Female Facial Expression
KDEF	Karolinska Directed Emotional Faces
KNN	K Nearest Neighbor
kTP	Knight Tour Pattern in a 3 x 3 neighborhood
KTP	Knight Tour Pattern in a 5 x 5 neighborhood
LBP	Local Binary Pattern
LBP-AW	Local Binary Pattern - Adaptive Window

LDA Linear Discriminant Analysis

LDDSCP Local Dominant Directional Symmetric Coding Patterns

LDMEP Local Directional Maximum Edge Patterns

LDN Local Directional Number

LDP Local Directional Pattern

LDSP Local Directional Structural Pattern

LDTerP Local Directional Ternary Pattern

LDTP Local Directional Texture Pattern

LGC Local Gradient Coding

LGC-HD Local Gradient Coding - Horizontal Diagonal

LIOP Local Intensity Order Pattern

LMeP Local Mesh Pattern

LNEP Local Neighborhood Encoded Pattern

LOOP Local Optimal Oriented Pattern

LPDP Local Prominent Directional Pattern

LTCP Local Triangular Coding Patterns

LTrP Local Triangular Patterns

MBS Multi-Biometric Systems

MDCBP Multi-level Directional Cross Binary Pattern

ML Machine Learning

MP Mesh Pattern

MSBP Multi-Stage Binary Patterns

MTP Mega Triangular Pattern

mTP Mini Triangular Pattern

MCP Modified Chess Pattern

MUG Multimedia Understanding Group

MPVS-Net Multi-Path Variation Suppressing Network

NIR Near Infra Red

NB Naive Bayes

NEDP Neighborhood Aware Edge Directional Pattern

NND Neural Network based Image Descriptor

OVA One Versus All

OVO One Versus One

PCANet Principal Component Analysis Network

PGBP Petersen Graph based Binary Pattern

PHOG Pyramid of Histogram of Oriented Gradients

PI Person Independent

PTP Positional Ternary Pattern

QUEST Quadrilateral Senary Bit

RADAP Regional Adaptive Affinitive Pattern

RAF Real World Affective Faces

RCP Radial Cross Pattern

RCSP Radial Cross Symmetric Pattern

ResNet Residual Network

RF Random Forest

RGB Red, Green and Blue

RGB-D Red, Green, Blue - Depth

RL Reinforcement Learning

RMP Radial Mesh Pattern

RP Radial Pattern

SAFL Self Adaptive Feature Learning

SBDP Square Based Diagonal Pattern

SBoFs Spatial Bag of Features

SERD Salient Expression Region Descriptor

SIFT Scale-Invariant Feature Transform

SOM Self-Organizing Map

SRC Sparse Representation based Classifier

SURF Speed Up Robust Transform

SR Super Resolution

SVM Support Vector Machine

TFEID Taiwanese Facial Expression Image Database

TL Transfer Learning

TN True Negative

TP	True Positive
VIS	Visual Light Scenario
VGG	Visual Geometry Group
VTS	Variation Training Set
WFBT-SBP	Weighted Full Binary Tree-Sliced Binary Pattern
WGFD	Windmill Graph based Feature Descriptors
WSEFEP	Warsaw Set of Emotional Facial Expression Pictures
YCbCr	Y is the brightness (luma), Cb is Blue minus Luma (B-Y) and Cr is Red minus Luma (R-Y)
YIQ	Luminance (Y), In-Phase (I) and Quadrature (Q)
YUV	Luminance (Y), Blue minus Luminance (U) and Red minus Luminance (V)

List of Notations

r_i	Pixels corresponding to Rook positions
b_i	Pixels corresponding to Bishop positions
k_i	Pixels corresponding to Knight positions
p_c	Center pixel
I	Hue
Q	Saturation
β	Query image
(M_l, N_l)	Labelled training instances
F^v	Feature vector
$X(:,:)$	Kernel function
α_l	Lagrange's multipler
ϕ	Bias parameter
γ	Number of classes
$s(u, v)$	Signum function used for comparing pixel intensities
(y, z)	Pixel intensities in a sample 5 x 5 neighborhood
n x n	Size of chess board
N x N	Input image size
$Hist$	Histogram
d	Distance
rd	Radius
fv	Feature vector
C x C	Block size
p	Number of neighbors

w	Weight matrix
fv_{cp}	Feature vector corresponding to CP
fv_{kTP}	Feature vector corresponding to kTP
fv_{KTP}	Feature vector corresponding to KTP
fv_{rmp}	Feature vector corresponding to RMP
fv_{rcp}	Feature vector corresponding to RCP
fv_{csp}	Feature vector corresponding to CSP
fv_{wgfd_h}	Feature vector corresponding to $WGFD_h$
fv_{wgfd_v}	Feature vector corresponding to $WGFD_v$
fv_{pgbp}	Feature vector corresponding to PGBP
fv_{mTP}	Feature vector corresponding to mTP
fv_{MTP}	Feature vector corresponding to MTP
fv_{ffnnd_1}	Feature vector corresponding to $FFNND_1$
fv_{ffnnd_2}	Feature vector corresponding to $FFNND_2$

Chapter 1

Introduction

Personal identity refers to a set of attributes (e.g., name, social security number etc.) that are associated with a person. The process of creating, preserving and destroying the individual identities is known as identity management. Person authentication is one of the most important activities in identity management, with the goal of determining or verifying an individual's identity claim. In general, a person can be recognized in three ways [1]: (i) Knowledge based (ii) Token based and (iii) Biometric based system. In Knowledge based system, a person is recognized based on "what he knows" (e.g., password, social security number etc.), whereas, in Token based system, a person is recognized based on "what he possesses" (e.g., passport, driving license etc.). In Biometric based system, a person is recognized based on "who he is" (physical traits) or "what he does" (behavioral traits) [2].

Formally, Biometric recognition can be defined as the science of establishing the identity of an individual based on the physical and behavioral traits of a person in a fully automated or semi-automated manner [3]. A biometric recognition system requires a person to be physically present at the time of authentication, thus preventing the need to remember a password or carrying a token. As a result, biometric traits cannot be easily lost, shared or duplicated [1, 3, 4]. Negative recognition and non-repudiation methods determine the individuals who have enrolled in a system, but later denies it and the individuals who have utilized a system but contradicts later. The passwords and Identity (ID) cards cannot provide negative recognition and non-repudiation operations. Whereas, negative recognition and non-repudiation are possible only with the help of biometrics.

Fingerprints, face, retina, iris, ear, palmprint, hand geometry etc. correspond to the physiological biometric traits. Keystroke dynamics, gait, signature, voice and speaker recognition etc. are the commonly used behavioral traits [3, 5]. All biometric traits satisfy the properties such as universality, uniqueness, permanence, collectability and circumvention [4, 6]. In today's world, biometric recognition has been applied in commercial applications (security applications, financial applications and mobile applications), legal applications (in the fields of justice and law enforcement), government applications (border control and airport applications, health care applications) and in tracking applications (screen navigation and aviation).

1.1 Biometric Recognition System

Biometric Recognition System (BRS) involves two phases namely enrollment phase and identification / verification phase [7]. The block diagram of an BRS is shown in figure 1.1. The five modules involved in an BRS are: sensor module, feature extractor module, template generator module, comparator module and decision module. The sensor module helps in acquiring the biometric characteristics of a person. At the time of biometric acquisition, there might be unwanted background information and noise. Hence, pre-processing techniques such as segmentation are performed for removing unwanted background information and noise removal is performed by applying filters. The feature extraction module plays a major role in an BRS as the classification is entirely dependent on the extracted features. Depending on the biometric trait and the type of application, the number of features extracted might be varied. When a person presents his biometric to the system for the first time, then it is known as enrollment phase. As a part of feature extractor module, the features are extracted from the captured biometric trait. The purpose of template generator module is to convert the extracted features into a template and store it in a database. The same process is repeated for feature extraction from the probe biometric at the time of identification / verification phase. The comparator module performs the comparison between the probe template and the reference / stored template and sends the result to the decision module. Based on the result, the decision module provides a match (accept) or a non-match

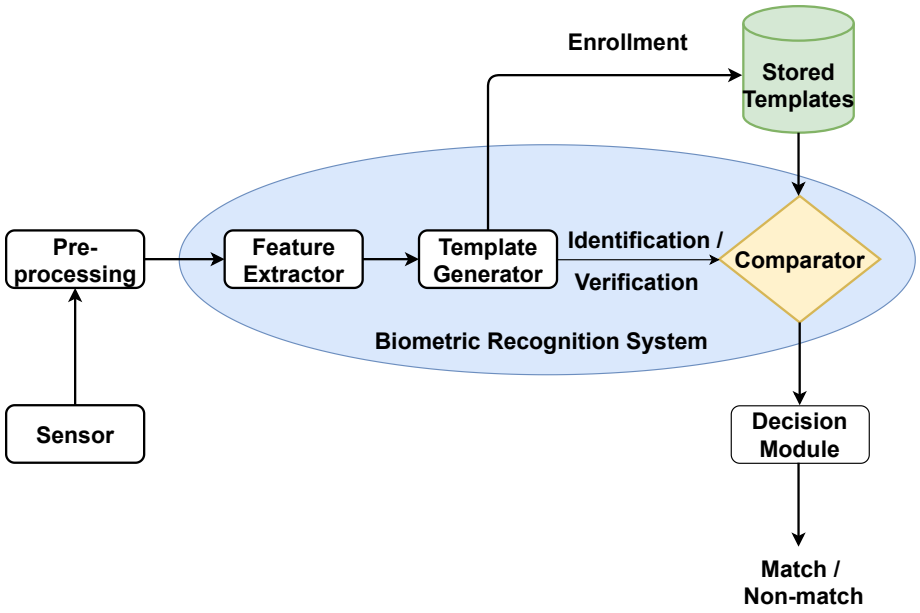


Figure 1.1: Block diagram of Biometric Recognition System

(reject) decision.

1.2 Face Recognition

Face recognition is an interesting area of research in computer vision and is applicable in a wide variety of domains such as in Automated Teller Machine (ATM), railway reservation systems, passport authentication, surveillance operations etc. Although, there are many biometric traits such as fingerprint, iris, gait, palmprint etc., face is chosen because of the following reasons:

- Face is the most common biometric used by human beings to recognize each other.
- Everyone has almost got a “fairly unique face”.
- The face of an individual can be identified even from a long distance. Therefore, recognition of a face is comparatively easier.
- Face data can be captured with contact less devices.
- No physical interaction is required on behalf of the user.

- Face recognition does not requires an expert to interpret the comparison.
- Face recognition can use existing hardware infrastructure.

Thus, by all these advantages, the topic of face recognition is chosen for further analysis and study.

1.2.1 Challenges in Face Recognition

Due to novel Corona virus, the companies across the globe started to move away from traditional fingerprint scanners and started adopting Artificial Intelligence (AI) based facial recognition technology. Although, face recognition is widely being deployed, there are still some challenges in face recognition as follows:

- Intra-personal variations
- Inter-class similarities
- Illumination variations
- Pose variations
- Facial Expressions
- Disguises
- Live Detection
- Occlusions
- Ageing
- Low resolution

Among all these challenges in face recognition, the area of Facial Expression Recognition (FER) is chosen for further analysis and study. It is because, human life starts with an emotion and ends with an emotion. In the life between birth and death, every person relates his current situation with any one of the emotions. Overall, life is an emotional journey

between the life and death of a person. Also, facial expressions are common across cultures and traditions. As, facial expressions are part and parcel of our lives, the area of FER is chosen for research and analysis.

1.3 Facial Expression Recognition

Communication involves using verbal and nonverbal cues to convey the intended meaning to others. In verbal communication, words and sounds are used to express our needs, intentions, thoughts and emotions to others [8]. Whereas, in non verbal communication, information is exchanged through gestures, facial expressions, eye contact, signs, body language, paralinguistics, haptics, proxemics etc. [9]. Of all these, facial expressions are of utmost importance as they can convey a lot of information about the unsaid internal mood and emotions of a person. The human face can display different feelings such as fear, happy, sad, surprise, disgust etc. which are universally understood and are common across different cultures and traditions [10]. Also, facial expressions provide valuable information about the person's attitude, internal emotions, personality and they also enable us to develop opinions of people surrounding us [11]. Facial expressions are considered crucial as they can compliment or contradict the information being conveyed through verbal words. Now a days, facial expressions are considered as soft biometrics and can act as a valuable supplementary biometric information to automated person identification systems [12, 13].

The automatic recognition of facial expressions has become a significant practical necessity as they have been incorporated in various real life applications such as Human Computer Interaction (HCI), deceit or lie detection, surveillance, affective computing, pain assessment, clinical psychology, robot control and behavioral profiling etc. Because of such wide range of applications and continuous evolution in these areas, Automated Facial Expression Recognition (AFER) has gained increased attention among the researchers in the recent years. The human beings have the inbuilt ability to recognize different facial expressions. But, for a system to recognize various expressions reliably and accurately, it is a tough task [14]. The various complexities such as spontaneous expressions, inconsistent acquisition conditions, ethnicity variations, illumination variations, aging factors,

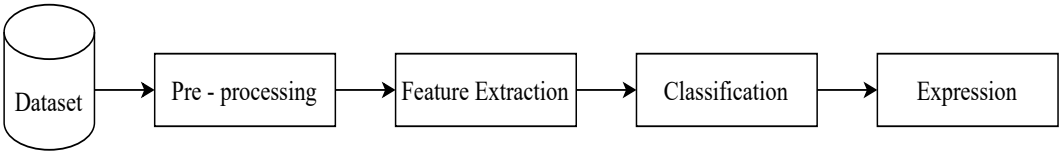


Figure 1.2: Basic structure of Facial Expression Recognition System

noise variations etc. are creating challenges for developing an AFER system [15].

1.3.1 Structure of Facial Expression Recognition System

FER system follows the classical pattern recognition tasks such as image acquisition, pre-processing, feature extraction and classification. Initially, for the input images from the dataset, pre-processing is performed for detecting the facial region. Next, feature extraction is performed to extract distinctive and valuable features from the facial images. The extracted features are then passed on to the existing classifiers such as K Nearest Neighbor (KNN), multi-class Support Vector Machine (SVM), Neural Networks etc [16] for appropriate expression classification. The basic structure of an FER system is shown in figure 1.2.

1.3.2 Feature Extraction

The performance of an FER system is mainly impacted by the method employed for feature extraction and the classification technique applied. Inadequately extracting the features might degrade the performance, even after using the best classification techniques [17]. So, it is essential to design an appropriate and reliable feature descriptor for enhancing the performance of an FER system. The techniques proposed for feature extraction in the literature can broadly be categorized as hand-crafted features, learned features and hybrid features [18]. Hand-crafted features are pre-designed for extracting relevant facial expressions and the learned features are obtained by making use of deep neural networks. In deep learning approaches [19–22], the appropriate features and classification weights are learnt collectively for recognizing the facial expressions. In case of hybrid features, the features obtained by two feature extraction methods are combined. Also, spatial and spatio-temporal representations are another way of classifying feature extraction techniques corresponding

to a single frame and a sequence of frames respectively [23]. The existing methods related to FER have been summarized in detail in Chapter 2. From the literature, the hand-crafted features used for extracting facial features can be broadly classified into two classes namely geometric based approaches [24–27] and appearance based approaches [11, 17, 28–37]. In this thesis, the main emphasis is on appearance based methods as the existing methods, [38, 39] have shown that the expressional changes typically occur on some main facial regions such as the neighborhood of the mouth, nose and eyes. This implies that details of local facial regions can be used to discriminate between expressions.

1.3.2.1 Geometric based Approaches

The geometric based approaches encode the locations, shapes, deformation, corners and contour information of facial components for characterising the facial structure [26, 27, 40, 41]. Although, these geometric features represent facial geometry, they fail to capture specific local information such as ridges, changes in skin texture etc. These geometric based methods need accurate tracking and detection of facial landmarks, which becomes difficult in different imaging conditions.

1.3.2.2 Appearance based approaches

In the literature, appearance based approaches are further classified into holistic (global) based approaches [36, 37, 42] and local based approaches [11, 17, 28–35]. The appearance based approaches represent the face image by applying image filters on the whole face (global) or on specific regions (local) for extracting appearance variations (e.g. wrinkles, skin changes) in facial images. Eigenfaces [36], Fisherfaces [37] and Linear Discriminant Analysis (LDA) [42] are some of the most widely used global based methods. As these global based methods are aimed at representing a facial image globally, they are unsuitable for capturing finer appearance changes corresponding to various facial expressions [31]. The research on local based methods focuses in two directions: texture based methods and edge based methods. The local texture based feature descriptors [11, 17, 28–30] detects gradient variations of an image for predicting various facial expressions, whereas, the edge based feature descriptors [31–35] uses filters to detect the edges, which are more relevant

in facial expressions.

A person's face can depict many expressions. There is minute difference between one expression to other expression, as conveyed by humans. In facial expression recognition, it is very important to capture those minute details related to expressions for accurate classification. From the literature study, the local appearance based approaches [17, 28–35, 43] have proven to be effective for facial feature extraction as they are aimed at examining the local regions for describing the various curvilinear features (curved and straight edges), corners etc. Also, the local texture based approaches are able to capture micro-level texture information such as specific skin changes, ridge details and minute characteristics that are more prevalent in facial expressions. Also, these local texture based methods can extract a detailed set of features which are noise resistant and they only need the class labels of the images for training purposes. The main strength of the local based methods is that, from the available images in the dataset, the relevant features can be extracted without requiring much training data and computing resources. Hence, in this work, local texture based feature descriptors have been proposed for facial feature extraction.

1.4 Overview of the Proposed Methods

Initially, the images from the benchmark FER datasets have been pre-processed using Viola Jones approach [44] for detecting and cropping the facial region from the entire facial image. Histogram equalization technique [45] is then applied for normalizing the illumination effects in an image. Upon the normalized images, the proposed feature extraction techniques have been applied and correspondingly the feature response maps have been obtained. The entire facial image could be divided into overlapping or non-overlapping regions for feature extraction, which could enhance the recognition performance of a system [46]. Hence, the feature response maps have been further divided into C x C non-overlapping blocks. Based on this block size, the feature response maps have been divided into 'T' regions. From each of these 'T' regions, features extracted are concatenated together to obtain a feature vector. The feature vectors have been generated using the proposed methods for both training and testing images and are given as an input to a multi-

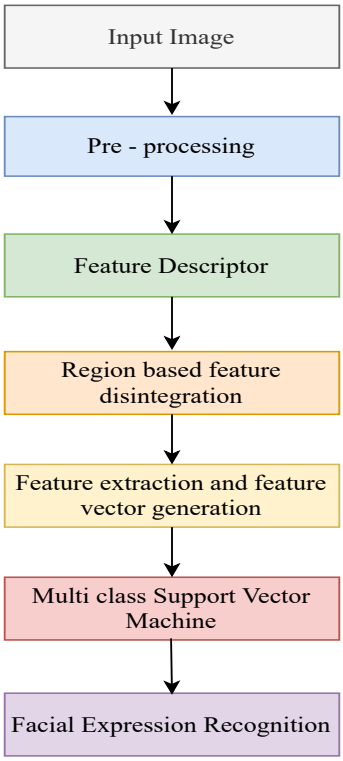


Figure 1.3: An overall flow of the proposed methods

class SVM for predicting the class labels of testing images. An overall flow of the proposed methods is shown in figure 1.3. Among all the stages mentioned in figure 1.3, feature descriptor stage plays an important role for accurate recognition of facial expressions. Hence, in this thesis, the main focus is on developing new feature descriptors for enhancing the recognition accuracy of an FER system.

1.5 Benchmark Datasets

Although, people regularly recognize many distinct emotions, for the most part, research studies have been limited to six basic categories. Anger, disgust, fear, happy, sad and surprise expressions correspond to the basic six expressions. Neutral and contempt are treated as the seventh and eighth expressions and embarrass and pride correspond to the ninth and tenth expressions. The experiments have been performed on ten FER datasets for validating the efficiency of the proposed feature descriptors. Among those ten datasets, eight belong to 'in the lab' datasets, one dataset (Real World Affective Faces (RAF)) belongs to 'in the

wild' category and the remaining dataset (Facial Expression Research Group (FERG)) is an animated facial expression dataset. The number of images considered for experimental evaluation across datasets is shown in table 1.1.

- Japanese Female Facial Expression (JAFFE) dataset [47], consists of 213 facial images belonging to seven facial expressions, collected from ten Japanese female subjects. For each facial expression belonging to a particular subject, there are almost four images present in the dataset. The image size is 256 x 256 pixels.

- Multimedia Understanding Group (MUG) dataset [48] has image sequences obtained from 86 subjects (51 male and 35 female). The image size is 896 x 896 pixels. The images corresponding to 45 subjects from those 86 subjects were selected for experimental evaluation. For each person in the dataset, there are five images corresponding to each expression.

- Extended Cohn-Kanade dataset (CK+) dataset [49] contains 593 image sequences captured from 123 subjects. Each sequence starts with a neutral expression and ends with the apex of an expression. The dataset has frontal facial images belonging to basic six expressions. The threw apex frames from each sequence are selected for each expression class [17]. The images are captured at 30 frames per second with images size being 640 x 480 (or) 640 x 490 pixels.

- OULU-CASIA dataset [50] has images captured from 80 subjects whose age lies in the range of 23 to 58 years old. The expressions were captured in both Near Infra Red (NIR) and Visual Light Scenario (VIS). Also, all expressions are recorded in strong, dark and weak environments. For each illumination, 480 video sequences were captured, so a total of 2880 video sequences are present in this dataset. The image size is 320 x 240 pixels. For the basic six expressions, the three peak frames from each expression are chosen, and the images for neutral expression were collected from the onset of each recording session [17].

- Taiwanese Facial Expression Image Database (TFEID) dataset [51] from National Yang-Ming Univeristy has 7200 stimuli collected from 20 male and 20 female sub-

Figure 1.4: Sample images from TFEID dataset. (a) Anger (b) Disgust (c) Fear (d) Happy (e) Neutral (f) Surprise (g) Sad.

jects. The image size is 480 x 600 pixels. In this dataset, there is almost one image per each expression. The sample images belonging to seven expressions for TFEID dataset are shown in figure 1.4.

- Karolinska Directed Emotional Faces (KDEF) dataset [52] is established by Karolinska Institute, Sweden. This dataset contains 4900 images obtained from 70 subjects (35 female and 35 male), whose age lies between 20 to 30 years old. The images have been captured for seven different facial expressions from the subjects. The images are captured twice from five different angles ($+90°$, $+45°$, $0°$, $-45°$, $-90°$) respectively with image size being 562x762 pixels. The frontal pose ($0°$) images have only been chosen for experimental evaluation.

- Warsaw Set of Emotional Facial Expression Pictures (WSEFEP) dataset [53] has 210 images collected from 30 individuals of which 14 are male and the remaining 16 are female. The image size is 800 x 542 pixels. The subjects have given poses for all seven facial expressions.

- Amsterdam Dynamic Facial Expression Set (ADFES) dataset is the full form of ADFES dataset [54]. This dataset has three more expressions namely contempt, embarrassment and pride expressions apart from the basic seven expressions. 216 images are present in this dataset. But, for experimental evaluation, only 215 images are considered. The image size is 720 x 576 pixels. The images were captured from 22 persons of which 12 are male and the remaining 10 are female whose age lies in between 18-25 years old.

- RAF dataset [55] contains 29,672 images captured from the real world. This dataset

Figure 1.5: Sample images from RAF dataset. (a) Anger (b) Disgust (c) Fear (d) Happy (Joy) (e) Neutral (f) Sad (g) Surprise

Figure 1.6: Sample images of Bonnie from FERG dataset. (a) Anger (b) Disgust (c) Fear (d) Happy (Joy) (e) Neutral (f) Sad (g) Surprise

has images belonging to 6 basic and 12 compound emotions. The image size is 100 x 100 pixels. For experimental analysis, only the images belonging to basic seven emotions are considered. The sample images belonging to seven expressions for RAF dataset are shown in figure 1.5. The number of training and testing images considered for experimental evaluation in RAF dataset are shown in table 1.2.

- FERG [56] has 55,767 annotated facial images from six stylized characters. MAYA software was used to create those six characters. Every character's images are divided into seven types of expressions. The image size is 256 x 256 pixels. The sample images belonging to seven expressions for FERG dataset are shown in figure 1.6. The number of images (subject wise) considered for experimental evaluation in FERG dataset are shown in table 1.3 and the number of training and testing images considered for experimental evaluation are shown in table 1.4.

1.6 Experimental Setup

For different 'in the lab' datasets, as a part of pre-processing, Viola Jones algorithm [44] is used for detecting the facial region and the images are then resized into 120 x 120 image

Table 1.1: Number of images considered for experimental evaluation across datasets

Expressions	**6 Expressions**						**7th Expression**	**8th Expression**	**9th & 10th Expressions**		
Dataset	**Anger**	**Disgust**	**Fear**	**Happy**	**Sad**	**Surprise**	**Neutral**	**Contempt**	**Embarrass**	**Pride**	**Total**
JAFFE [a]	30	29	32	31	31	30	30	-	-	-	213
MUG [b]	225	225	225	225	225	225	225	-	-	-	1575
CK+ [c]	135	177	75	207	84	249	369	-	-	-	1296
OULU [d]	240	240	240	240	240	240	240	-	-	-	1680
TFEID [e]	34	40	40	40	39	36	39	68	-	-	336
KDEF [f]	70	70	70	70	70	70	70	-	-	-	490
WSEFEP [g]	30	30	30	30	30	30	30	-	-	-	210
ADFES [h]	22	22	22	22	22	21	22	21	21	20	215
RAF [i]	867	877	355	5957	2460	1619	3204	-	-	-	15339
FERG [j]	9169	8571	7419	7331	7627	8712	6938	-	-	-	55767

[a] https://zenodo.org/record/3451524#
[b] https://mug.ee.auth.gr/fed/
[c] https://sites.pitt.edu/~emotion/ck-spread.htm
[d] https://www.v7labs.com/open-datasets/oulu-casia
[e] https://bml.ym.edu.tw/tfeid/modules/wfdown-loads/
[f] https://www.kdef.se/download-2/register.html
[g] http://www.emotional-face.org/wsefep
[h] https://aice.uva.nl/research-tools/adfes-stimulus-set/adfes-stimulus-set.html
[i] http://www.whdeng.cn/raf/model1.html#dataset
[j] http://grail.cs.washington.edu/projects/deepexpr/ferg-2d-db.html

Table 1.2: Training and testing images in RAF dataset

	Anger	**Disgust**	**Fear**	**Happy**	**Neutral**	**Sad**	**Surprise**	**Total**
Training	705	717	281	4772	2524	1982	1290	12271
Testing	162	160	74	1185	680	478	329	3068

Table 1.3: Distribution of images in FERG dataset

Subject	**Anger**	**Disgust**	**Fear**	**Joy**	**Neutral**	**Sad**	**Surprise**	**Total**
Aia	1644	1596	1116	1232	1200	1469	1746	10003
Bonnie	2059	2088	1255	1438	1243	1506	1682	11271
Jules	1479	1321	1858	1317	1344	1475	1961	10755
Malcolm	1100	1190	1018	1089	1083	1034	1065	7579
Mery	1428	911	1035	1140	941	1045	1058	7558
Ray	1459	1465	1137	1114	1128	1098	1200	8601
Total	9169	8571	7419	7330	6939	7627	8712	55767

Table 1.4: Training and testing images in FERG dataset

	Anger	**Disgust**	**Fear**	**Joy**	**Neutral**	**Sad**	**Surprise**	**Total**
Training	8169	7571	6419	6331	5938	6627	7712	48767
Testing	1000	1000	1000	1000	1000	1000	1000	7000

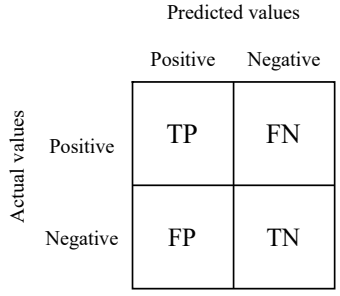


Figure 1.7: Confusion matrix for two class classification

responses. Then, histogram equalization method is applied for normalizing the illumination levels in the images. For experimental evaluation, Person Independent (PI) scheme is followed. As a part of PI scheme, leave one subject out policy (for all datasets except CK+) has been followed, i.e at each time, one subject is excluded from training and is used for testing. In case of CK+ dataset, for each subject, as the number of images for each expression are not balanced, ten fold PI cross validation is performed. Thus, by excluding a subject in this manner ensures person independence. In RAF dataset, there are 12,271 training and 3068 testing facial images and an image size of 100 x 100 pixels is considered for experimental evaluation. In FERG dataset, out of 55,767 images, 48,767 images have been used for training, and the remaining 7,000 images (1000 from each expression, chosen randomly) have been used for testing purposes and an image size of 48 x 48 pixels is considered for experimental evaluation.

For the purpose of classification, a multi-class SVM has been utilized. The performance of the proposed methods has been analysed in terms of recognition accuracy and confusion matrix, which are discussed in section 1.6.1. The experiments have been performed using MATLAB R2018a tool on i5 processor with Windows 10 operating system and 16 GB RAM. For comparison analysis, some existing variants of binary patterns have been implemented in our setup and correspondingly the recognition accuracy is reported. Following PI approach, the recognition accuracy is computed by taking the mean of accuracy obtained from each subject / fold. The results generated by the existing variants of binary patterns implemented in our environment setup might be different from the accuracy reported in the papers, because of the different image size and block size considered for experimental evaluation. For methods other than the variants of binary patterns, the comparison results are directly taken from their corresponding papers.

1.6.1 Performance Measures

The following measures have been used to assess the performance of an FER system:

- **Confusion Matrix**: Confusion matrix is a combination of both actual and predicted values. A sample confusion matrix is shown in figure 1.7. The following terminol-

ogy is followed for confusion matrix: True Positive (TP), False Positive (FP), False Negative (FN), True Negative (TN).

- **Recognition Accuracy**: Recognition accuracy is defined as the number of correctly classified instances out of the total instances. The corresponding equation for calculating recognition accuracy is shown in eq.(1.1).

$$Recognition\ Accuracy = \frac{Number\ of\ Correctly\ classified\ instances}{Total\ number\ of\ instances} \tag{1.1}$$

1.7 Motivation, Aim & Objectives

Facial expressions can reveal the genuine intentions of a person. In social interaction, humans understand facial expressions and respond accordingly. In the same manner, can systems understand facial expressions and respond accordingly? To achieve this task, a system should recognize the facial expression correctly and then interpret the meaning of it. So, the main emphasis is on accurately recognizing the facial expressions. For accurate expression recognition, feature extraction stage plays a major role as the performance of an FER system mostly depends on the extracted features. If insignificant features are extracted, even the best classification techniques may fail to classify accurately [17]. Developing new feature extraction techniques based on the knowledge obtained from previous methods can help in improving the performance of a system. But, the most important problem in FER system lies in extracting significant and discriminative patterns from the facial images.

In general, image processing applications require both global and local features to be extracted for efficient classification. But, in case of facial expression recognition, different expressions are portrayed on the same facial image. For experimental evaluation, as majority of 'in the lab' datasets have been considered, they are mostly captured under controlled environment. From one expression image to another expression, there are only minute differences and those differences need to be effectively captured for accurate classification. Convolutional Neural Network (CNN)'s can be used to automatically extract the features from images. But, majority of the existing CNN based FER methods only extract the fea-

tures from the entire facial image. In other words, the traditional CNN based methods may not fully exploit the recognition-effective information encoded in expressional images.

The existing methods [38, 39] have shown that the expressional changes typically occur on some main facial regions such as the neighborhood of the mouth, nose and eyes. This implies that details of local facial regions can be used to discriminate between expressions. Also, among those appearance based methods, the texture based methods [17, 57, 58] have been found to be suitable to extract valuable features such as ridge details, specific skin changes and minute characteristics within a local region. Also, the local texture based approaches need only the class labels of the images for training purposes. For accurately detecting facial expressions, the neighboring pixel's relationship with the reference pixel is also essential for detecting finer appearance changes with respect to various expressions. Recently, facial expressions have been applied to provide advanced level of security to biometric systems. "The correct recognition of feeling can enhance the security of biometric system. For example, a recognized expression of fear can be used to decline entry in a secured area even if the face ID is recognized correctly" [59]. Thus, facial expressions are currently deployed for providing advanced level of security to the existing biometric systems. This motivated us towards proposing new local texture based feature descriptors for enhancing the recognition accuracy of FER systems.

1.7.1 Aim

This dissertation aims to provide some texture based feature descriptors for FER systems intended to improve overall recognition accuracy.

1.7.2 Objectives

The main objectives of this dissertation are stated as follows:

- To study and investigate the existing feature descriptors proposed for FER systems.
- To design and develop effective feature descriptors for FER systems.
- To study and investigate different weights for feature extraction.

- To utilize the concepts of feature level fusion for maximizing the recognition accuracy.

1.8 Overview of the Contributions in the Thesis

In this thesis, the following contributions related to feature extraction methods have been made for FER systems.

1. By utilizing the concept of chess game rules, a local feature descriptor named Chess Pattern (CP) has been proposed for extracting the facial features in a local neighborhood.

2. Inspired by the Knight tour problem in a n x n Chess board, novel feature descriptors named Knight Tour Pattern in a 3 x 3 neighborhood (kTP) and Knight Tour Pattern in a 5 x 5 neighborhood (KTP) have been proposed for facial feature extraction.

3. Radial Mesh Pattern (RMP), a combination of Radial Pattern (RP) and Mesh Pattern (MP) has been proposed for overcoming the limitations of existing feature descriptors such as Local Binary Pattern (LBP), Local Mesh Pattern (LMeP) and CP.

4. Three feature descriptors namely Radial Cross Pattern (RCP), Chess Symmetric Pattern (CSP) and Radial Cross Symmetric Pattern (RCSP) have been proposed for overcoming the limitations of CP, Local Gradient Coding (LGC) and it's variants.

5. Proposed Windmill Graph based Feature Descriptors (WGFD) for facial feature extraction by drawing inspiration from the Windmill Graph.

6. Proposed Petersen Graph based Binary Pattern (PGBP) for facial feature extraction by drawing inspiration from the Generalized Petersen Graph.

7. Proposed Local Triangular Patterns (LTrP) for facial feature extraction by drawing inspiration from the shape of a Triangle.

8. Proposed Feed Forward Neural Network Structure Inspired Feature Descriptors (FFNND) for facial feature extraction by drawing inspiration from the structure of a feed forward neural network.

1.9 Thesis Organization

The rest of the Chapters in this thesis have been organized in the following manner: In Chapter 2, the related and the relevant state-of-the-art methods in the field of FER systems have been described. Towards the end of this Chapter, the research findings from the literature studies have been summarized.

In Chapter 3, local texture based feature descriptors namely CP, kTP and KTP inspired by the chess game rules and Knight tour problem have been presented for facial feature extraction in a local neighborhood. Both, CP and KTP methods consider 5x5 neighborhood for extracting the multi-level information in a local neighborhood. In this thesis, the concept of feature level fusion has been adopted in all of the works to achieve maximum recognition accuracy.

In Chapter 4, Modified Chess Pattern (MCP) feature descriptors have been described in detail. RMP method is a combination of RP and MP. RCP, CSP and RCSP have been proposed to overcome the limitations of some existing methods such as CP, LGC and it's variants. Here, in this Chapter, the experiments have been conducted on each of these feature descriptors independently with different weights to find out the optimal recognition accuracy.

In Chapter 5, texture based feature descriptors namely $WGFD_h$, $WGFD_v$, LTrP and PGBP inspired by shapes of Windmill graph, Generalized Petersen Graph (GPG) and Triangle graph have been proposed for feature extraction in a local neighborhood. The experiments have been conducted using $WGFD_h$ and $WGFD_v$ methods independently with different weights to determine the optimal recognition accuracy. For LTrP and PGBP methods, only the concept of binary weights is used, as six bits are only involved for extracting each feature.

In Chapter 6, two hand-crafted feature descriptors namely $FFNND_1$ and $FFNND_2$, in-

spired by the structure of a feed forward neural network have been proposed for facial feature extraction. The proposed $FFNND_1$ and $FFNND_2$ methods have been modelled in such a manner to capture the adjacent pixel relationships in a local neighborhood.

In Chapter 7, the concluding remarks of the thesis and the suggestions for further analysis and study have been reported.

Chapter 2

Literature Survey

In this Chapter, the existing methods related to face detection have been reported in section 2.1. The existing techniques related to feature extraction in the literature have been mentioned in section 2.2. The types of information fusion in biometrics have been reported in section 2.3. The various types of classifiers used for facial expression recognition analysis have been summarized in section 2.4. Towards the end of this Chapter, in section 2.5, the research findings analyzed from the literature studies have been summarized.

2.1 Face Detection

Face detection is a significant phase of FER systems. In an image, the face region is detected using the facial features such as skin color, texture, edge and face muscle motion. These characteristics help in distinguishing a facial region from the background. The input image is segmented into two parts during this phase: one representing the face region and the other representing a non-face region. There are numerous face detection methods available in the literature such as the Eigenspace method, Adaptive Skin Color method, Viola–Jones method [44] and their algorithms are developed based on the Haar classifier, Adaboost, and Contour Points [60].

2.1.1 Eigenspace Method

Pentland et al. [61] proposed Eigenspace method for locating the face under variable poses. Also, modular Eigenspace descriptors have been used for recognizing the facial image with salient features. Essa et al. [62] used the Eigenspace method for locating the face in any random image sequence. The Eigenfaces define the subspace of sample images as a face space [63]. The distance between the observed image and face space was estimated using the projection coefficients and the signal energy for detecting the presence of a face in an image. Similarly, the spatio-temporal filtering method was used for detecting a face in an image sequence. The concept of thresholding was applied to the filtered image, for causing binary motion that aids in the analysis of 'motion blobs' over time. Each motion blob represents a human head for detecting the location of the face.

2.1.2 Adaptive Skin Color Method

Skin color is a useful feature for detecting faces [64, 65]. The most commonly used color systems are Red, Green and Blue (RGB), Cyan, Magenta and Yellow (CMY), Luminance (Y), In-Phase (I) and Quadrature (Q) (YIQ), Luminance (Y), Blue minus Luminance (U) and Red minus Luminance (V) (YUV), Y is the brightness (luma), Cb is Blue minus Luma (B-Y) and Cr is Red minus Luma (R-Y) (YCbCr). Any one of the color systems is preferred based on color dependency. For color intelligence, YIQ and YUV color systems are commonly used [66]. The components I and Q in the YIQ color model refer to hue and saturation, respectively, where I denotes the value of face skin color in YIQ space that changes in a specific range between 30 and 100. The equation to calculate I is shown in eq.(2.1).

$$I = 0.596 * R - 0.274 * G - 0.322 * B \tag{2.1}$$

Simultaneously, the hue range of face skin color in YUV space is between 105° and 150°. YIQ and YUV color systems are synthesized to create a primary face skin-color model. If an image meets the following conditions, $30 \leq \mathrm{I} \leq 100$, $105^{\circ} \leq \theta \leq 150^{\circ}$, then it is considered as a skin color. The majority of studies use skin color for face detection based

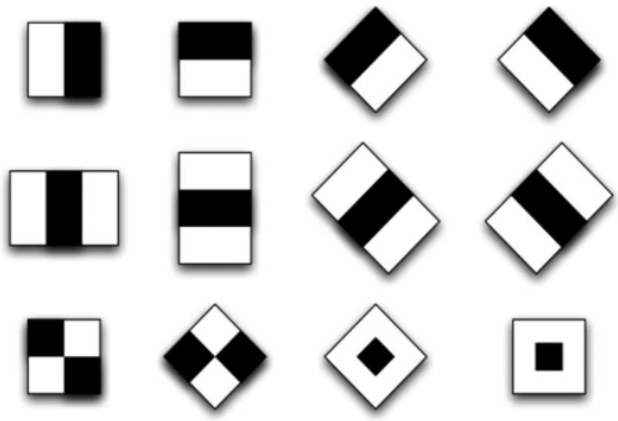


Figure 2.1: Haar features considered for feature extraction

on a fixed threshold scheme, which results in large errors due to illumination and pose variations. To obtain the actual face region that satisfies the face geometric pattern, an iterative thresholding algorithm has been proposed [67]. However, due to it's high computational cost, it is unsuitable for real-time applications. Cho et al. [68] introduced an adaptive skin color filter that uses a linear discriminant function to separate the skin region from a complex background by adaptively adjusting the threshold values. To influence illumination and pose variations, the gamma corrective method is used. Zhao-yi et al. [66] proposed an adaptive skin color and structure model for multi-pose color images in a complex background that greatly improves accuracy while effectively ignoring the impact of illumination levels.

2.1.3 Haar Classifier Method

In a real-time environment, the Haar classifier is considered as a reliable face detection method [14]. Haar features are considered for detecting facial edges, motions, lines and skin color. Haar features are a black and white connected rectangular box, that are used for feature extraction, as shown in figure 2.1. Haar features are easily scaled, and positions are examined by increasing or decreasing the pixel intensities at various parts of an image. The value of the located feature is the difference between the sum of pixels in the black and white regions of the rectangle box [69]. During the training phase, the Haar classifier detects the features that contribute to face detection problems. As a result, the computational

cost and complexity in the testing phase are reduced, resulting in a high detection accuracy.

2.1.4 Adaboost Method

The AdaBoost is an ensemble approach for face detection [70, 71] that is widely used due to it's improved accuracy and low computational complexity. It is a well-known face detection method that has a low false positive rate. Adaboost's main limitation is it's sensitivity to noisy data and outliers [70]. To eliminate negative samples, a set of image features is trained with several classifiers in cascade using Adaboost. The first classifier's output will be used as input to the next classifier, which will be used to obtain an accurate face region. As a result, a strong classifier was created, which aids in reducing the number of features and, as a result, results in high detection accuracy. Kheirkhah et al. [65] proposed a color and complex image-based hybrid and robust face detection system. This hybrid method uses both skin color information and Adaboost-based face detection and gives better performance with minimum execution time.

2.1.5 Contour Points

Face detection using contour points improves accuracy [72, 73]. In an image sequence, the first pixel of the first frame is scanned based on skin color, and that point is considered as the head's first contour point. In the same way, the remaining contour points in a frame are computed. The pixel under consideration is referred to as the seed point. The contour point's direction is set with the identified seed point, and the detection path can be clockwise or anti-clockwise. Face motion is detected when there is a shift in two consecutive frames in a sequence and a shift in contour points above a threshold [72]. Aniruddha et al. [73] used a contour-based method to detect and track the human face in video frames. To accomplish the task of getting proper face contour, logical operations and Gaussian filters have been employed. To detect and track the face in an image sequence, the scalar and vector distances of a rectangular window drawn from four corner points of two consecutive frames are calculated.

2.2 Feature Extraction

The techniques proposed for feature extraction in the literature can broadly be categorized into hand-crafted features, learned features and hybrid features [74]. Hand-crafted features are pre-designed for extracting relevant facial expressions, whereas, the learned features are obtained by making use of deep neural networks. The hybrid features usually refer to the features, obtained by combining two or more of the feature extraction methods [75]. The classification of feature extraction methods related to FER is shown in figure 2.2. In the literature, the hand-crafted features are again categorized into geometric based and appearance based features [17].

2.2.1 Hand-crafted Features

2.2.1.1 Geometric based Features

The geometric based approaches encode the locations, shapes, deformation, corners and contour information of facial components for characterising the facial structure [26, 27, 76]. From the facial image, fiducial points were extracted by Zhang et al. [40] as landmark points for extracting geometrical features, by modelling the shape and locations information. Valstar et al. [41] proposed a method for tracking the facial landmark points and for detecting the action units in a facial image. Based on these detected action units in an image, the facial expressions can be recognized. In figure 2.3, an illustration of the geometric information from facial landmarks in the characterization of facial images is shown. Zangeneh et al. [77] considered the first and the last images and extracted differential geometric features from the important points of the face from those two images. Oztel et al. [78] detected the keypoints from facial images by focusing on eye and eyebrows and generated a feature set by using geometric relationships among those detected key points. For capturing the deformities caused by the movement of facial muscles due to various facial expressions, Sharma et al. [79] proposed a system that utilized the facial landmark points and generated three feature sets for determining the relative distances between the facial features.

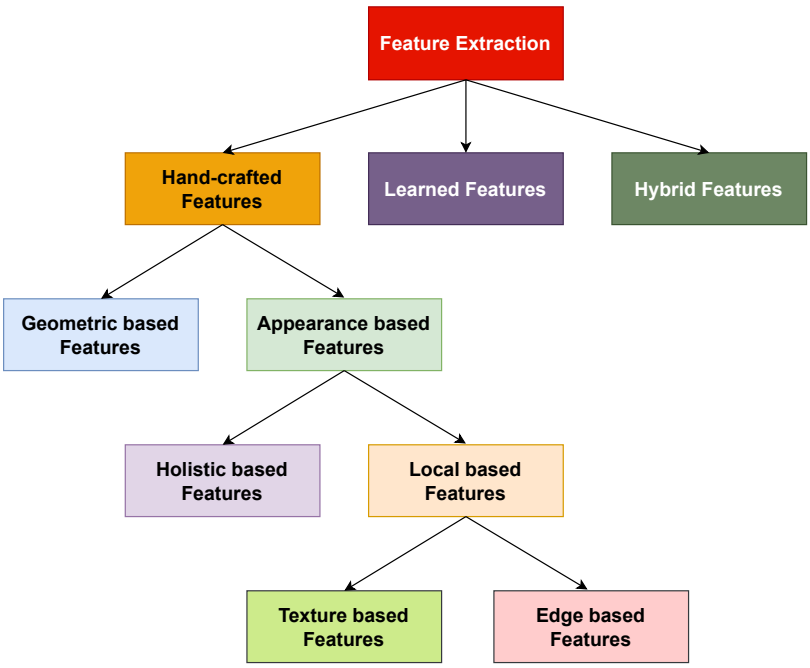


Figure 2.2: Classification of feature extraction techniques

Jain et al. [80] proposed Square Based Diagonal Pattern (SBDP) method on geometric model called Geometric Appearance Models (GAM) for extracting in-depth information from Red, Green, Blue - Depth (RGB-D) images. Chouhayebi et al. [81] detected the facial landmarks using Dlib library and extracted geometric features by considering the spatial positions between the landmarks. The features encoded by these geometric based approaches can describe the entire face image using a lesser number of features, which are scale and rotation invariant. Although, these geometric features represent facial geometry, they fail to capture specific local information such as ridges, changes in skin texture etc. These geometric based methods need accurate tracking and detection of facial landmarks, which becomes difficult in different imaging conditions. Also, these geometric features require additional pre-processing techniques for localizing various face components before the process of feature extraction.

2.2.1.2 Appearance based Methods

In the literature, appearance based approaches are further classified into holistic (global) based approaches [36, 37, 42] and local based approaches [11, 17, 28–35]. The appear-

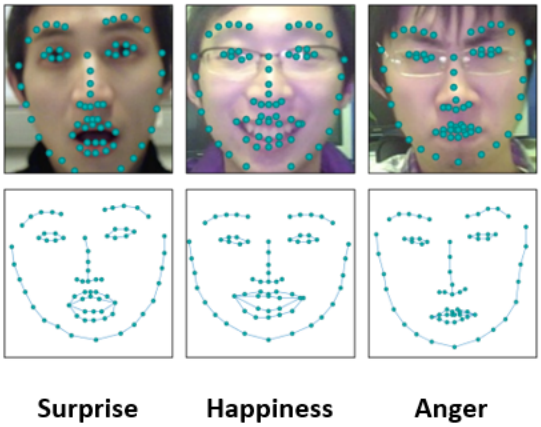


Figure 2.3: Illustration of the geometric information from facial landmarks in the characterization of facial images for FER

ance based features represent the face image by applying image filters on the whole face (global) or on specific regions (local) for extracting appearance variations (e.g. wrinkles, skin changes) in facial images.

2.2.1.2.1 Holistic based Features

Holistic-based methods consider the face as a whole and do not explicitly divide the face into sub-regions. Eigenfaces [36], Fisherfaces [37], LDA [42] and Information Discriminant Analysis (IDA) [82] are some of the most widely used holistic based methods. As these holistic based methods are aimed at representing a facial image globally, they are unsuitable for capturing finer appearance changes corresponding to various facial expressions [31].

2.2.1.2.2 Local based Features

LBP [46] is the most popular texture based method for facial feature extraction. LBP is computationally efficient and is also invariant to changes in monotonic illumination. In cases of intensity fluctuations, random noises, and non-monotonic illumination levels, the feature extraction capability of LBP is affected [83]. The process of calculating LBP code for a sample 3 x 3 image patch is shown in figure 2.4. Lai et al. [28] proposed Center Symmetric Local Binary Pattern (CSLBP) for greatly reducing the feature vector

dimensionality of LBP. In CSLBP, appropriate threshold needs to be chosen in prior from experimental analysis and the information related to center pixel is also neglected. Tong et al. [84] proposed LGC in a 3 x 3 neighborhood by encoding the gradient information in horizontal, vertical and diagonal directions for generating a feature vector. Kung et al. [85] proposed Dual Non-negative Graph Embedding (DSNGE) for efficiently representing the facial images using identity and expression subspaces. Sun et al. [86] presented Individual Free Representation Based Classification (IFRBC) that utilized the concepts of Variation Training Set (VTS) and Virtual VTS for remitting the side-effect caused by individual variations. Arshid et al. [87] presented a method for FER by drawing inspiration from compressive sensing theory and multi-resolution approach.

Arshid et al. [88] proposed Multi-Stage Binary Patterns (MSBP) for handling imbalance and local illumination by considering both gradient difference and sign difference. Verma et al. [89] proposed Quadrilateral Senary Bit (QUEST) Pattern for encoding the intensity changes in a local neighborhood by dividing the surrounding pixels and reference pixel into two quadrilaterals. Swapna et al. [90] proposed a system named 'Anubhav' for extracting the features only from active salient patches, which carry expression specific information. In Regional Adaptive Affinitive Pattern (RADAP) [17], an adaptive global threshold is generated for capturing the global and local invariant features in the local neighborhood. RADAP uses the multi-distance information for capturing the expression specific changes. Also, XRADAP, ARADAP and DRADAP operators are obtained from RADAP by performing Xor, Adder and Decoder operations respectively. Yang et al. [91] proposed Center Symmetric Local Gradient Coding (CSLGC) by considering the directional gradients in four directions in a center symmetric manner and used the average of the gradients for reducing the sensitivity to noise.

In Center Symmetric Local Octonary Pattern (CSLOP) [30], the gray value of neighboring pixels is compared with center pixel, also the gray values of four pairs of center-symmetric pixels is compared. By using CSLOP, feature vector length is greatly reduced and avoids selecting any threshold for comparison unlike CSLBP. The gradient feature map is obtained by finding magnitudes in the horizontal and vertical directions. To this generated feature map, CSLOP operator is applied for the purpose of feature extraction. The

features obtained are then fused with CSLOP for enhanced performance. Kas et al. [92] proposed Multi-level Directional Cross Binary Pattern (MDCBP) for texture recognition by combining both multi-radius and multi-orientation information. Drawing inspiration from human vision system, Sadeghi et al. [93] proposed a method for expression recognition based on gabor filters. Kumar et al. [94] proposed Weighted Full Binary Tree-Sliced Binary Pattern (WFBT-SBP) for analyzing an RGB image based on inter-pixel similarity patterns.

Kola et al. [57] proposed fusion of features obtained from singular values and Wavelet Based Local Gradient Coding - Horizontal Diagonal (LGC-HD) operator for effective facial expression recognition. Durga et al. [16] presented a new variant of LBP method named Local Binary Pattern - Adaptive Window (LBP-AW) that considers four neighbors and diagonal neighbors separately in a local neighborhood and utilized the Adaptive Window concept for extracting noise robust facial features. Subhadeep et al. [95] proposed Gammadion Binary Pattern of Shearlet Co-efficients (GBPSC) for illumination and noise invariant face recogntion. Arya et al. [58] proposed Local Triangular Coding Patterns (LTCP), which utilizes a set of pixels in a triangular neighborhood for extracting texture features from an image. Verma et al. [96] proposed Cross-Centroid Ripple Pattern (CRIP) for encoding the image features by using inter radial ripples.

The main property of local edge based (dense) feature descriptors such as Local Directional Number (LDN) [31], Local Directional Texture Pattern (LDTP) [32], Local Directional Pattern (LDP) [97], Local Directional Ternary Pattern (LDTerP) [33], Local Directional Structural Pattern (LDSP) [34], Angled Local Directional Pattern (ALDP) [35] are extracting the micro-edge level primitives present in the facial images due to the movements of facial muscles. In the local neighborhood, these edge based methods apply Kirsch compass masks [97] in the eight possible directions and measures the edge responses that are prominent and encode a few numbers to those specific directions. In LDP [97], the input image is convolved with Kirsch masks, and the top three values are assigned the value as one. As Kirsch masks are operated on a local 3x3 neighborhood, the presence of noise or intensity distortions in the local regions might affect the calculations of Kirsch value responses, which in turn may lead to assigning different code values for the patterns which

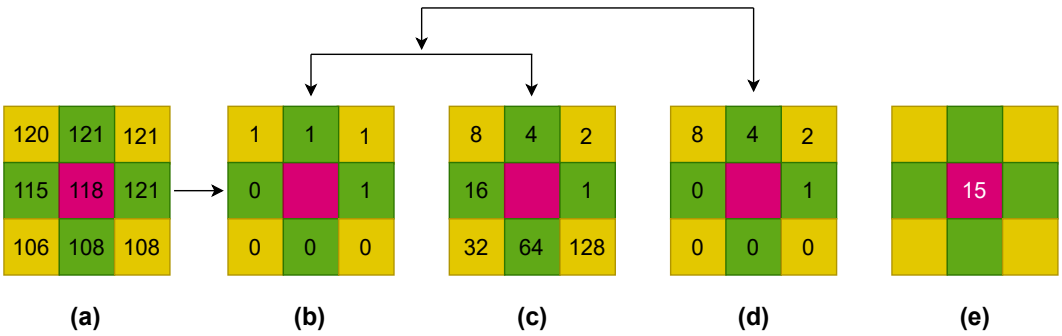


Figure 2.4: LBP code computation. (a) Sample 3 x 3 image patch (b) Thresholding (c) Weighting (d) Binary to Decimal conversion (e) Computed LBP

are similar and also similar code values can be assigned to various different patterns. Also, LDP cannot differentiate the changes in intensity on both the sides of an edge, which is necessary for obtaining crucial information such as lower or upper edges of lip, wrinkles and eyebrows etc. For overcoming this drawback, LDN [31] was proposed, which encodes both the directions of top negative and positive Kirsch response values. Even after preserving top 'k' positive and negative Kirsch responses, LDN is still affected by the noise prevalent in the local neighborhood.

Positional Ternary Pattern (PTP) [98] also encodes the directions based on the top two Kirsch responses, however, PTP aims at differentiating pixels in a flat region based on a pre-determined threshold and PTP fails for the images taken in various imaging and lighting conditions. In LDTP [32], the principal directional numbers are obtained in eight different directions and the difference in intensity values of opposing pixels in the principal directions is coded as numbers. Thus, LDTP uses both structural and contrast information for generating feature codes. In LDTerP [33], ternary patterns are proposed for extracting the emotion related features information and also a two-level grid was developed for differentiating the finer and coarse features. The finer grids are employed for capturing highly expression specific features and the coarser grids are employed for capturing features related to a non-expression. In the smoother portions of an image, both LDTP and LDTerP generates unstable patterns.

At every pixel, Neighborhood Aware Edge Directional Pattern (NEDP) [10] generates feature codes by considering the gradients at the neighbors with regard to the pre-defined

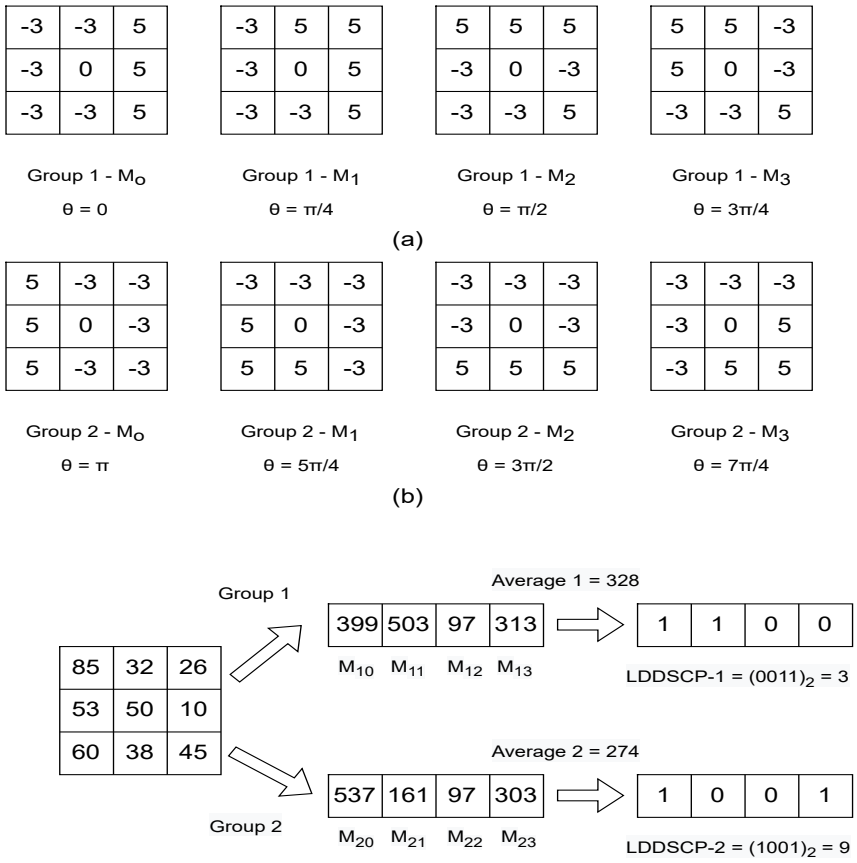


Figure 2.5: Process of feature extraction through LDDSCP. Two groups of Kirsch compass masks (a) first group (b) second group. Process of feature extraction through LDDSCP (c) LDDSCP code computation.

template orientations. In ALDP [35], the Kirsch responses are obtained by convolving an input image with Kirsch masks. From these Kirsch responses, the angular vector components in horizontal ($0°$), vertical ($90°$), diagonal($45°$) and anti-diagonal ($135°$) directions are calculated. In LDSP [34], the positional information of top two Kirsch mask responses is computed for extracting the structural information of a local neighborhood. In this LDSP, the top Kirsch response denotes the primary edge direction and the pixel's structure is denoted by the structural feature code. This LDSP uses a global adaptive threshold for filtering out flat patterns, but the drawback is that threshold needs to be known in prior for stable feature extraction.

Local Prominent Directional Pattern (LPDP) [15] considered the statistical information in a local region for encoding meaningful edge based features despite some positional variations. Local Dominant Directional Symmetric Coding Patterns (LDDSCP) [99] have been proposed for effectively encoding the dominant directions of facial textures. The process of feature extraction using LDDSCP is shown in figure 2.5. In Local Optimal Oriented Pattern (LOOP) [11], the Kirsch responses are sorted and the assignment of weights is done as per the sorted responses, rather than using sequential weights. Maheswari et al. [100] proposed Local Directional Maximum Edge Patterns (LDMEP) that considered only the dominant magnitude and orientation directions information for better extracting facial information.

2.2.2 Learned Features

Most recent works in FER are aimed at applying deep learning based techniques for accurately classifying the images into various expressions. Aneja et al. [56] proposed DeepExpr, a transfer learning technique to map expressions from humans to animated characters. Hasani et al. [20] developed a 3D inception-resnet network for capturing the spatial and temporal relationships in the static images and videos. Li et al. [55] proposed Deep Locality Preserving Convolutional Neural Network (DLPCNN) to preserve the locality closeness by maximizing the inter-class scatters. Ji et al. [101] proposed a fusion network based on intra category common and distinctive feature representation. Xie et al. [102] presented the Deep Attentive Multi-Path Convolutional Neural Network (DAMCNN), which combines

Salient Expression Region Descriptor (SERD) with the Multi-Path Variation Suppressing Network (MPVS-Net).

Wu et al. [103] proposed Adaptive Feature Mapping (AFM) for transforming the feature distribution of testing samples into that of training samples. Zhao et al. [104] proposed an instance based transfer learning approach with multiple feature representations. Feutry et al. [105] proposed a framework to learn anonymized representation of statistical data. Minaee et al. [106] proposed Attentional Convolutional Network (ACN) model with less than ten layers for classifying emotions from facial images. Sun et al. [107] adopted Dictionary Learning Feature Space (DLFS) for training and Sparse Representation Classification for finding the emotion of query images. Verma et al. [108] proposed variants of Hybrid Inherited Feature Learning Network (HiNet) for capturing the local contextual information of expressive regions. Alenazy et al. [109] applied Gravitational Search Algorithm (GSA) for optimizing the parameters in Deep Belief Network (DBN). Li et al. [110] presented a framework based on Reinforcement Learning (RL) that contains two modules namely image selector for selecting useful images and a rough emotion classifier module that acts like a teacher for training image selector.

Xie et al. [111] proposed two branch Disentangled Generative Adversarial Network with two independent branches for processing facial and expressional information seperately. Li et al. [110] used RL for selection of relevant images for expression classification. Mohan et al. [112] proposed Facial Expression Recognition-Network (FER-net), an CNN for efficiently extracting the features from facial regions and the features extracted are passed on to the softmax classifier for identifying facial expressions. Saurav et al. [113] proposed Dual Integrated Convolutional Neural Network (DICNN) model for recognizing 'in the wild' facial expressions on embedded platform. Reddy et al. [114] proposed Deep Cross Feature Adaptive Network (DCFA-CNN) for extracting both high level responses as well as minute variations from a facial image. Chirra et al. [115] proposed a deep multi block CNN based ensemble method for extracting features from stylish, virtual and human characters. Fan et al. [116] proposed Hierarchial Scale CNN for extracting the information from kernel, network and knowledge scales. There are other existing CNN based techniques proposed recently like Visual Geometry Group (VGG) [22], Principal Compo-

nent Analysis Network (PCANet) [117], Residual Network (ResNet) [118], Self Adaptive Feature Learning (SAFL) [119], Image Filter based Subspace Learning (IFSL) [120] and Auxiliary Classifier Generative Adversarial Network (AC-GAN) [121] that have shown significant improvements in the field of FER systems.

2.2.3 Hybrid Features

Happy et al. [126] extracted both the shape (Pyramid of Histogram of Oriented Gradients (PHOG)) and appearance features (LBP) from the active facial patches. Majumder et al. [39] proposed a deep learning based hybrid method by combining geometric features with LBP features using autoencoders for improving the FER performance. Farooq et al. [127] extracted one dimensional R-transform features along with Self-Organizing Map (SOM) model on time sequential facial images. Ali et al. [122] presented a method based on Histogram of Oriented Gradients (HOG) algorithm characteristics and Sparse Representation based Classifier (SRC) for classifying the facial expressions with poses. Kalsum et al. [124] proposed a hybrid feature descriptor method by combining Spatial Bag of Features (SBoFs) with Scale-Invariant Feature Transform (SIFT) features and SBoFs with Speed Up Robust Transform (SURF) features. In table 2.1, summary of some feature extraction methods related to FER have been presented.

Wang et al. [125] proposed a hybrid feature representation by combining both SIFT and deep features and utilized SVM for classification. Javad et al. [128] presented a multi-stream CNN along with three hand-crafted features for improving the performance of FER with limited training data. Sen et al. [129] presented an idea of combining both texture based and geometric based features and utilized Directed Acyclic Graph Support Vector Machine (DAGSVM) for classifying the facial expressions. Li et al. [130] proposed a new set of salient patterns at facial key point locations and extracted both geometric and textural features for performing facial expression classification. Jeong et al. [131] proposed Deep Spatio Temporal Network (DJSTN) by combining both appearance and geometric networks with joint fusion classifier. Wang et al. [132] presented Multi-parameter fusion feature Space (Multi-block LBP and HOG features) for representing the facial expressions and also used a Decision Voting strategy (MSDV) based on Nearest Neighbor classifier for

Table 2.1: Summary of some feature extraction methods

Feature	Strength	Limitations
LBP [46]	Computationally simple with high discriminating power and is invariant to grayscale changes.	Affected by image rotation and captures limited structural information.
LDTP [32]	Considers both structural and contrast information for generating feature codes.	In the smoother portions of an image, LDTP generates unstable patterns.
HOG [122, 123]	Has a capacity to provide fine-grained details in small scales and global information in large scales. It is invariant to photometric transformation and illumination changes.	Extraction takes more time as final feature vector grows larger.
SIFT [124, 125]	Invariant to illumination changes and affine rotation.	Affected by image rotation, computationally intensive and susceptible to high dimensional complexity.
Learned [55, 101]	Efficient descriptor.	Computationally intensive, requires high computing resources and large volume of data.
Hybrid [39, 126]	Features compliment each other.	Prone to computational complexity.

predicting the facial expressions.

Gogic et al. [133] proposed an algorithm by combining gentle boost decision trees for extracting local binary features around facial landmark points and used shallow neural networks for classification. Shanthi et al. [134] demonstrated an approach for analyzing the relation between the adjacent pixels and proposed a feature level fusion technique that combined both LBP and Local Neighborhood Encoded Pattern (LNEP). Liu et al. [135] proposed a hybrid feature extraction network by combining both the pixel level features and deep geometric features for enhancing the discriminating power of emotional features. For extracting discriminative facial features, Liu et al. [136] proposed a hybrid facial feature representation method by combining PHOG, Canny edge detector and LBP methods. Kalsum et al. [123] proposed fusion of global (HOG) and local based feature descriptors (Local Intensity Order Pattern (LIOP)) for facial emotion recognition.

2.3 Information Fusion in Biometrics

Biometric fusion is the processing of biometric modalities using several methods. Multi-Biometric Systems (MBS) are categorized into six types namely multi-instance, multi-sample, multi-sensor, multi-algorithm, multi-modal & hybrid systems [137]. In multi-instance, multi-sample, multi-sensor and multi-algorithmic fusion, a single biometric is used for fusion. In multi-modal systems, multiple biometric modalities are used for fusion. Multiple instances of the same biometric data are used for fusion in multi-instance systems. In multi-sample systems, several samples of the same modality captured at different times are fused together (e.g., left, right and frontal profiles of a face). In multi-sensor systems, data collected from multiple sensors are fused together. Different algorithms are employed to extract features from a single biometric modality and the information from all the desired feature sets are fused together in multi-algorithm systems (e.g., fusion of texture related features + geometric based features for FER). In multi-modal systems, the data obtained from several modalities are fused together [138]. Chang et al. [139] used the term 'hybrid' for describing systems that integrate a subset of the five types of MBS. For example, a hybrid system can be both multi-algorithmic as well as multi-modal in it's design.

Fusion can be performed at various levels such as sensor level, feature level, decision level or score level [138]. If fusion is performed prior to matching, then it is known as early fusion. If fusion is performed after matching, then it is known as late fusion [137]. Sensor level and feature level fusion comes under early fusion, whereas, decision level and rank level fusion comes under late fusion. The studies from the literature reveal that feature level fusion provides better recognition accuracy than other levels of fusion [140].

2.4 Classifiers

2.4.1 Support Vector Machine

SVM is a Machine Learning (ML) algorithm mainly used for classification / regression problems [17, 141]. In between two classes, SVM attempts to find an optimal hyperplane that maximizes the inter-class distance [142]. The main strength of SVM lies in handling complex non-linear data and being robust to overfitting. Suppose $\{(M_l,N_l), l=1,....,q\}$ be the labelled training instances, for $M_l \in F^v, N_l \in \{-1,+1\}$, where F^v be the feature vector corresponding to each expression. A new query (test) image (β) is classified using eq.(2.2) as:

$$Q(\beta) = sign(\sum_{l=1}^{q} \alpha_l N_l X(M_l, \beta) + \phi) \tag{2.2}$$

where, X(:, :) corresponds to a kernel function, α_l corresponds to Lagrange's multipliers, and ϕ corresponds to the bias parameter.

2.4.2 K-Nearest Neighbor

KNN [143–145] is an instance-based learning algorithm that performs classification or regression using a non-parametric technique. The training data is made up of vectors in a multidimensional feature space, each labelled with a class. The algorithm's training phase consists solely of storing these feature vectors and the classes to which they belong. The classification step predicts an input feature or set of features by assigning the class with the closest features to the input. Euclidean Distance (ED) and Hamming Distance (HD)

are the two popular distance metrics for determining which features are closest features to the input [146]. The main strength of KNN lies in the simplicity of implementation and in quick training. However, it necessitates a large amount of storage space, testing is slow, sensitive to noise and performs poorly with high-dimensional data. Another issue with this classification / regression approach is that unbalanced classes can lead to inaccurate predictions (classes with more samples usually dominate the predictions, even if incorrectly). Setting class weights is one solution to this problem.

2.4.3 Naive Bayes

Naive Bayes (NB) classifiers [147–149] are a family of probabilistic ML classifiers, based on the Bayes theorem that assumes strong independence between the features. The following equation expresses the Bayes theorem:

$$P(A|B) = \frac{P(B|A)(A)}{P(B)} \tag{2.3}$$

The equation used to calculate the likelihood of event A occurring given that B is true is shown in eq.(2.3). However, because these classifiers assume that features are independent, the NB classifier will not correlate features when making a prediction in an FER system. This is undesirable because there are obvious correlated features when making facial expressions, such as when one is surprised, where the mouth and the eyes are clearly correlated. However, the benefits of this classifier lies in simple implementation and it's ability to scale well for large datasets.

2.4.4 Hidden Markov Model

The Hidden Markov Model (HMM) [150–152] is a probabilistic model capable of predicting a sequence of unknown variables based on a set of observed variables. In an FER system, for example, this would imply predicting happiness (hidden variable) based on a smile (observed variable). The strength of this classifier is it's ability to model arbitrary features from observations, it's ability to merge multiple HMMs to classify more data, and

it's incorporation of prior knowledge into the model. This classifier, however, is computationally expensive and suffers from overfitting.

2.4.5 Decision Tree

As a classifier, Decision Tree (DT) [87, 153, 154] is essentially a flowchart represented as a tree model. An DT classifier divides the database into smaller sets of data until no more splits are possible, and the resulting leaves are the classification classes. This classifier's strength include the ability to learn nonlinear data relationships, the ability to handle high-dimensional data, and the ease of implementation. The main drawback of this classifier is overfitting, because it can continue branching until it memorises the data during the training step.

2.4.6 Random Forest

Random Forest (RF) [155–157] is essentially an ensemble classifier, consisting a group of DTs. Each DT produces a prediction, and the final prediction is based on majority voting, which means that the most predicted class is the last prediction. It has the advantage of reducing overfitting over just one DT, because, it reduces bias by averaging the ensemble predictions. RF, however, has the disadvantage of becoming slower as it's complexity increases (e.g., by adding more DTs to the ensemble).

2.4.7 Sparse Representation Classifer

SRC [122, 158] constructs a dictionary from training facial images, with dictionary elements corresponding to facial Action Units (AU). If there are enough training samples for each AU class, the test AU can be represented as a linear combination of only those training samples that belong to the same AU class. Facial image differences [159], Gabor wavelet coefficients [160] and other facial representation features [161] could be used to define the dictionary in a variety of ways. Typically, the SRC is designed for AU detection in sub-regions of the facial image or in facial difference images. Sparse representation could be carried out in a robust manner that is resistant to partial occlusions.

2.4.8 Convolutional Neural Network

CNN is a type of neural network that is primarily used in Computer Vision (Deep Learning (DL)) due to it's ability to solve multiple image classification problems [102, 110]. CNN's can even outperform humans in some of these tasks because they can detect and identify underlying patterns that the human eye cannot detect. An input image is processed by the CNN's hidden layers, which decompose it into features. These features are then used for classification, typically via a Softmax function that selects the class with the highest probability from the probability distribution as the predicted class. Different problems require different CNN models and need fine- tuning techniques to achieve a high classification accuracy. This is mainly caused due to over-fitting / under-fitting problem. Over-fitting and under-fitting problems can be solved by using various ways:

- By adding more layers to increase the model complexity.
- By using dropout layers [162] to randomly disable a set of nodes during training to avoid the model memorising patterns rather than learning them.
- By tuning the model's parameters during training, such as epochs, batch size, learning rate, and class weight etc.
- By increasing the training data set by adding more samples by using Data Augmentation (DA) techniques.
- Transfer Learning (TL) can be used when the database is too small. TL employs a pre-defined model that has already been trained on a large database, and it can be fine-tuned for it's own classification problem using a smaller database.

2.5 Summary

In this Chapter, the existing techniques proposed for face detection, feature extraction and classification have been presented. Also, the various types of fusion in biometrics has been reported. Based on the literature studies, the edge based methods were prone to be less

discriminating for different image regions, and in smoother regions of an image, the edge based approaches tend to generate unstable patterns. In the presence of noise, the edge based descriptors are strongly inconsistent and are often influenced by minor local distortions. In general, the output of an CNN, i.e., the feature response maps of the last layer, will be considered as the high-level semantic concept for representing the input. However, majority of the existing CNN based FER methods only extract the features from the entire facial image. These methods emphasise the importance of a facial expression while ignoring the information about the local details. In other words, traditional CNN based methods may not fully exploit the recognition-effective information encoded in expressional images. Also, the number of training images, batch size, image size, learning rate and number of model parameters have a significant impact on the performance of a neural network.

In this thesis, face detection is done using Viola Jones method. As benchmark 'in the lab' FER datasets contains less number of images, the data available for training in those datasets is limited. In facial expressions, the difference between two expressions is too small, and it is essential to to capture those finer appearances properly for accurate expression recognition. From the literature, local texture based approaches have proven to be useful for extracting such minute details from the facial images. Hence, in this thesis, the texture based feature descriptors have been proposed for extracting significant features from the local facial regions that are discriminative for expressional recognition. For the purpose of classification, a multi-class SVM classifier has been considered, as SVM is the most widely classifier in the field of ML and pattern recognition. In this thesis, for Chapters 3 and 4, a multi-class classifier model employing $\gamma\ (\gamma - 1)/2$ binary SVM models with One Versus One (OVO) approach and linear kernel function has been followed, where γ corresponds to the total number of classes. For Chapters 5 and 6, a multi-class SVM with both OVO and One Versus All (OVA) approaches and linear kernel function have been followed.

Chapter 3

Chess Game Rules Inspired Feature Descriptors

A brief introduction about feature extraction techniques and the importance of appearance based methods is mentioned in section 2.2.1.2. The main contributions of this Chapter are summarized as:

- CP, a game rules based feature descriptor has been proposed with an intention to capture significant facial features in a local neighborhood by aiming to generate different feature codes for corner, edge and flat image regions.
- Inspired by the Knight's tour problem in graph theory, new feature descriptors named Knight Tour Patterns (kTP and KTP) have been proposed for extracting the facial features in a local neighborhood.
- To the proposed feature descriptors, apart from binary weights, different weights such as fibonacci, prime, natural, squares and odd have been applied to determine the optimal recognition accuracy.

A brief description about the chess game is presented in section 3.1. In section 3.2 CP, a game rules based feature descriptor has been discussed. In section 3.3, kTP and KTP, inspired by the Knight tour in graph theory have been described in detail. In section

3.4, the experimental results corresponding to CP, kTP and KTP have been presented and analyzed. In section 3.5, the contributions in this Chapter have been summarized.

3.1 Chess Game

Chess is generally a two-player game played on a 8 x 8 chessboard with 64 squares of alternating colors. In a chess board, the light colored squares correspond to the 'white' and dark colored squares correspond to the 'black'. The horizontal and vertical rows in a chess board are called as ranks and files respectively. There are sixteen pieces from six types available for each player. Each piece in a chess board moves in a distinct way. The moves of each piece are described below.

- King $\rightarrow$ A King can move exactly one square in vertical, horizontal or diagonal directions as long as no piece is blocking his path.
- Queen $\rightarrow$ A Queen can move any number of vacant squares in vertical, horizontal or diagonal directions.
- Rook $\rightarrow$ A Rook can move any number of vacant squares in vertical or horizontal directions.
- Bishop $\rightarrow$ A Bishop can move any number of vacant squares in diagonal directions only.
- Knight $\rightarrow$ A Knight can jump to any square in 'L' shape either by moving two squares vertically and one square horizontally (or) by moving two squares horizontally and one square vertically. It is the only piece that can jump over a piece (either our own or opponent's).
- Pawn $\rightarrow$ If a square is vacant, a Pawn moves one square straight forward. A Pawn can also advance two squares straight forward if it hasn't moved yet, as long as both squares are empty.

Among these six chess pieces, only the moves of Rook, Bishop and Knight are considered in this thesis. In general, Rook corresponds to Elephant, Bishop corresponds to Camel and

Black						
White						
Piece	King	Queen	Rook	Bishop	Knight	Pawn

Figure 3.1: Six types of black and white chess pieces available in a chess game

Knight corresponds to Horse. In figure 3.1, the six types of 'black' and the 'white' chess pieces are shown.

3.2 Chess Pattern

The Chess Pattern (CP) was initially proposed by Tuncer et al. [163] for texture recognition. CP is a local texture based image descriptor, which is based on the movements of chessmen such as Rook, Bishop and Knight in a 5 x 5 neighborhood. Motivated by it's success in the field of texture recognition, the same CP methodology has been adopted into the field of biometrics, towards addressing the PI FER problem. Instead of using 3 x 3 neighborhood, the proposed CP considers 5 x 5 neighborhood in an image. A sample representation of a 5 x 5 block (B) at a pixel location (y,z) is shown in eq.(3.1). The center pixel in 5 x 5 block is denoted by pixel (p_c), is shown in eq.(3.2).

$$B = \begin{bmatrix} y,z & y,z+1 & y,z+2 & y,z+3 & y,z+4 \\ y+1,z & y+1,z+1 & y+1,z+2 & y+1,z+3 & y+1,z+4 \\ y+2,z & y+2,z+1 & y+2,z+2 & y+2,z+3 & y+2,z+4 \\ y+3,z & y+3,z+1 & y+3,z+2 & y+3,z+3 & y+3,z+4 \\ y+4,z & y+4,z+1 & y+4,z+2 & y+4,z+3 & y+4,z+4 \end{bmatrix} \tag{3.1}$$

$$p_c = B_{y+2,z+2} \tag{3.2}$$

In a 5 x 5 neighborhood, with reference to the pixel (p_c), the possible positions where

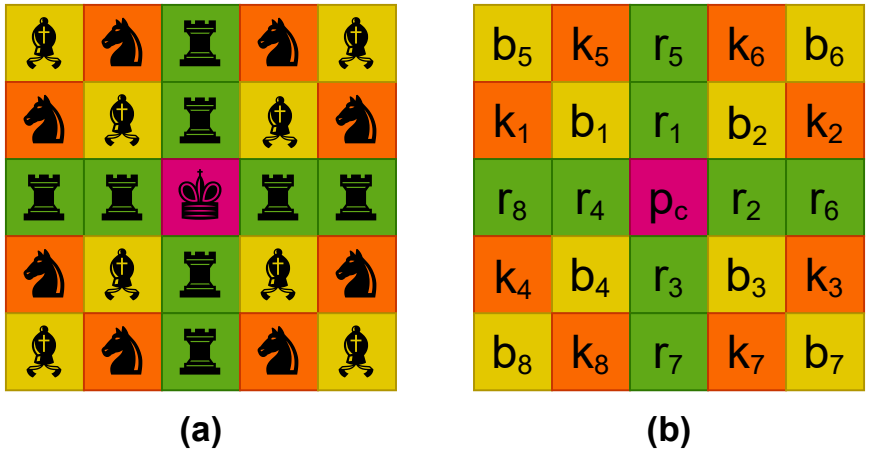


Figure 3.2: (a) The represented chessmen on the 5 x 5 block (b) CP obtains features using the numbering scheme given to these chessmen

Rook, Bishop and Knight could move are determined and are systematically shown in figure 3.2(a). In figure 3.2, to maintain consistency with chess game terminology, Rook, Bishop and Knight are denoted by r, b and k respectively. Initially, the possible positions of Rook, Bishop and Knight are determined in the 3 x 3 neighborhood, followed by the 5 x 5 neighborhood. The possible positions are named sequentially in a clockwise manner, starting from the north direction, as shown in figure 3.2(b). The existing feature descriptors, such as LBP, LDP, LDN and PTP generates the same feature codes for corner, edge and flat image regions. To overcome this particular drawback, CP, a local feature descriptor has been proposed for facial feature extraction. CP method has been modelled by considering neighborhood pixel relationship for extracting Rook, Bishop and Knight features and by considering the adjacent pixel relationship for extracting Rook_Knight, Rook_Bishop and Bishop_Knight features.

3.2.1 Feature Extraction through CP

Feature extraction includes extracting six patterns based on the positions of Rook, Bishop and Knight in a 5 x 5 neighborhood. The movements of Rook captures both vertical and horizontal texture information, whereas Bishop captures the diagonal information and Knight captures the information from the remaining leftover pixel positions. The pixel positions corresponding to the moves of Rook, Bishop, Knight, Rook_Knight, Rook_Bishop,

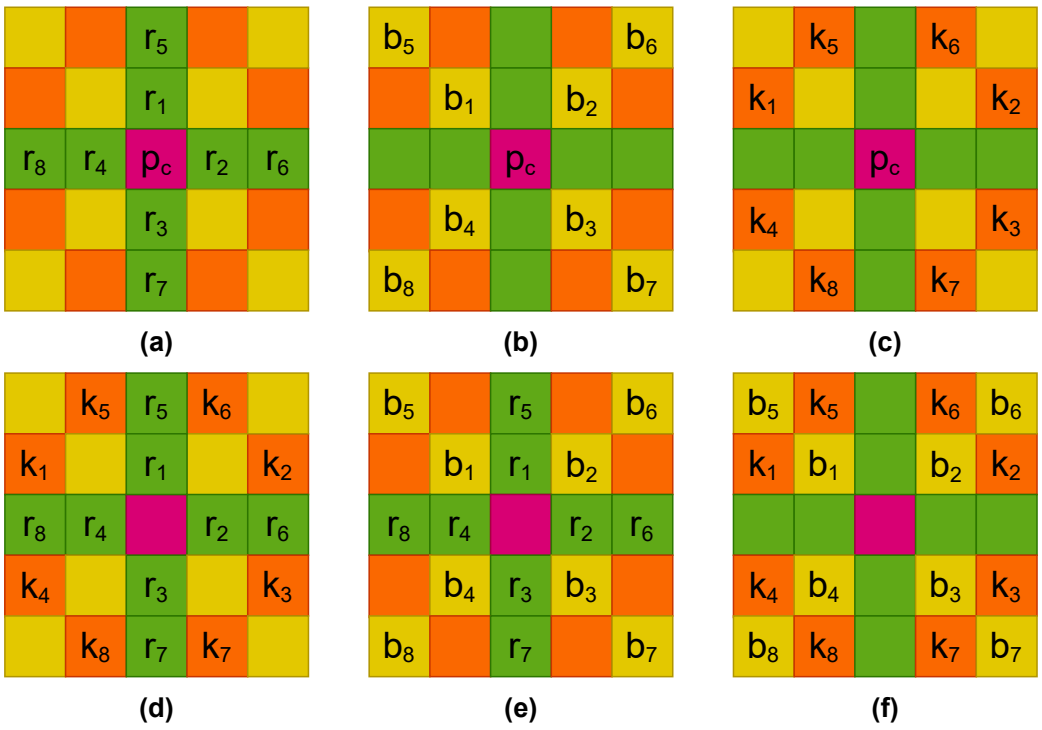


Figure 3.3: The possible positions where (a) Rook (b) Bishop (c) Knight (d) Rook_Knight (e) Rook_Bishop and (f) Knight_Bishop are placed in a 5 x 5 block

Knight_Bishop in a 5 x 5 neighborhood are shown in figure 3.3. As the positions of three chessmen (Rook, Bishop, Knight) are considered, the features are extracted using the following formula of $3C_1 + 3C_2 = 6$. The first three features extracted are: Rook, Bishop and Knight. The numbered pixels $r_{1,2,...,8}$, $b_{1,2,...,8}$, $k_{1,2,...,8}$ are compared sequentially with pixel (p_c) for extracting Rook, Bishop and Knight features respectively. If the result is greater than or equal to zero, then the corresponding bit is encoded as one, else it is encoded as zero. The resultant binary number thus formed is multiplied by weight vector (w_x). The corresponding equations for calculating Rook, Bishop and Knight features are shown in the eqs.(3.3) to (3.6).

$$Rook = \sum_{i=1}^{8} (s(r_i, p_c) * w_x) \tag{3.3}$$

$$s(u, v) = \begin{cases} 1, & \text{if } u - v \geq 0 \\ 0, & otherwise \end{cases} \tag{3.4}$$

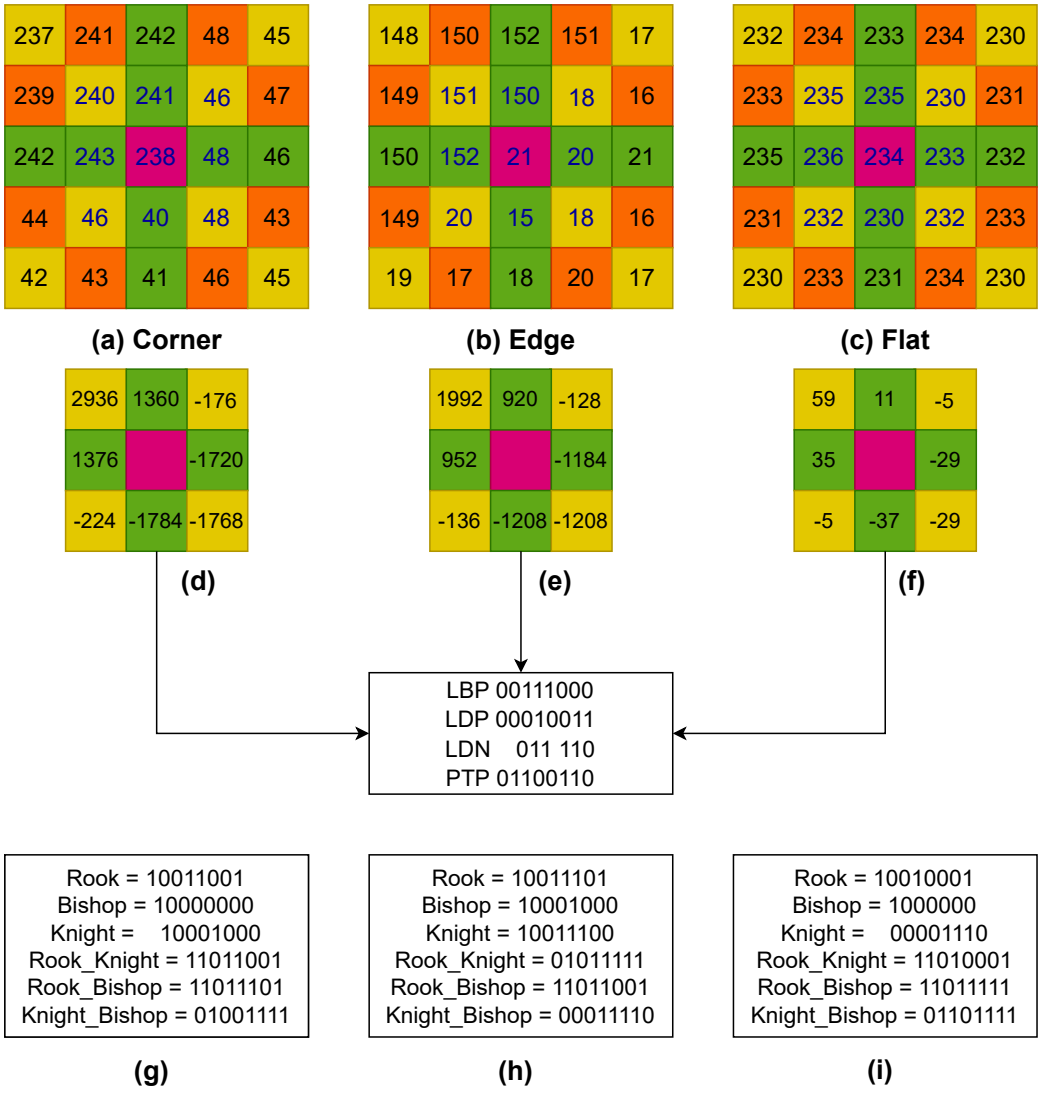


Figure 3.4: Example for drawbacks of existing feature descriptors. Same feature codes are generated for (a) corner (b) edge (c) flat regions. (d-f) Kirsch responses for patterns (a-c) (g-i) Proposed CP generating almost different codes for different edge patterns (a-c)

$$Bishop = \sum_{i=1}^{8}(s(b_i, p_c) * w_x) \tag{3.5}$$

$$Knight = \sum_{i=1}^{8}(s(k_i, p_c) * w_x) \tag{3.6}$$

$$Rook\_Knight = \sum_{i=1}^{8}(s(r_i, k_i) * w_x) \tag{3.7}$$

$$Rook\_Bishop = \sum_{i=1}^{8}(s(r_i, b_i) * w_x) \tag{3.8}$$

$$Knight\_Bishop = \sum_{i=1}^{8}(s(k_i, b_i) * w_x) \tag{3.9}$$

Next, the features are calculated by combining two chessmen. So, the combinations of Rook_Knight, Rook_Bishop and Knight_Bishop features are considered. Here, the numbered pixels in the first chessmen are compared with the numbered pixels of second chessmen, rather than comparing with pixel (p_c). Thus, Rook_Knight feature is obtained by comparing the pixel intensities in locations $r_{1,2,..,8}$ with $k_{1,2,..,8}$, as indicated in eq.(3.7). Similarly, Rook_Bishop feature is obtained by comparing the pixel intensities in locations $r_{1,2,..,8}$ with $b_{1,2,..,8}$, as indicated in eq.(3.8). In the same manner, Knight_Bishop feature is obtained by comparing the pixel intensities in locations $k_{1,2,..,8}$ with $b_{1,2,..,8}$, as indicated in eq.(3.9). If the result is positive, then the corresponding bit is encoded as one, else it is encoded as zero. The resultant binary number thus formed is multiplied by w_x. Finally, the feature vector of CP is obtained by horizontally concatenating the features obtained from all the six extracted features. Usually, binary weights are used in the calculation of feature vector. Hence, the feature vector length corresponding to CP is 1536 (256 x 6 = 1536). The feature codes generated by the existing methods such as LBP, LDP, LDN, PTP and proposed method, CP for different image portions such as corner, edge and flat are shown in figure 3.4. From the figure 3.4, an observation can be made that, two feature codes are common between 3.4(g) and 3.4(h). Similarly, there is one feature common between 3.4(g) and 3.4(i). The existing methods generate only a single feature code, whereas, CP method generates six features. By generating more features, the different regions of an image can be

(a) (b) (c) (d) (e) (f) (g)

Figure 3.5: (a) Happy expression image from TFEID dataset. Feature response maps generated by (b) Rook (c) Bishop (d) Knight (e) Rook_Knight (f) Rook_Bishop and (g) Knight_Bishop features using binary weights.

easily distinguished. But, accommodating six features increases the feature vector length. With an intention to maximize recognition accuracy and to effectively reduce the feature vector length of CP, different weights such as fibonacci [164], prime, natural, squares and odd weights have been applied to the CP method, which are discussed in the subsection 3.2.2.

$$x = \{binary, fibonacci, prime, natural, squares, odd\} \tag{3.10}$$

$$binary = \Big[128,\ \ 64,\ \ 32,\ \ 16,\ \ 8,\ \ 4,\ \ 2,\ \ 1\Big] \tag{3.11}$$

$$fibonacci = \Big[21,\ \ 13,\ \ 8,\ \ 5,\ \ 3,\ \ 2,\ \ 1,\ \ 1\Big] \tag{3.12}$$

$$prime = \Big[19,\ \ 17,\ \ 13,\ \ 11,\ \ 7,\ \ 5,\ \ 3,\ \ 2\Big] \tag{3.13}$$

$$natural = \Big[8,\ \ 7,\ \ 6,\ \ 5,\ \ 4,\ \ 3,\ \ 2,\ \ 1\Big] \tag{3.14}$$

$$squares = \Big[64,\ \ 49,\ \ 36,\ \ 25,\ \ 16,\ \ 9,\ \ 4,\ \ 1\Big] \tag{3.15}$$

$$odd = \Big[15,\ \ 13,\ \ 11,\ \ 9,\ \ 7,\ \ 5,\ \ 3,\ \ 1\Big] \tag{3.16}$$

3.2.2 Different weights for feature extraction

Whenever, binary weights are used for feature extraction, the feature vector length is increased drastically. As feature vector length increases, the number of comparisons required and the computation time required also increases and feature reduction needs to be done for selecting the best features. So, instead of proposing a feature descriptor with high dimen-

Table 3.1: Different weights considered for feature extraction

Weights	Range	Maximum value	Length(fv)
Binary	0-255	255	256
Fibonacci	0-54	54	55
Prime	0-77	77	78
Natural	0-36	36	37
Squares	0-204	204	205
Odd	0-64	64	65

sions, and then applying feature reduction techniques, an alternative can be thought of to extract features with low dimensions. Based on this idea, different weighting schemes such as using fibonacci [164], prime, natural, squares and odd weights have been proposed and applied to CP method for facial feature extraction, which are shown in eqs.(3.10) to (3.16). Thus, w_x can take any one of the weight values among binary, fibonacci, prime, natural, squares and odd. Whenever, binary weights are used for feature extraction, the maximum possible value in calculation of feature vector is 255. Alternatively, whenever, fibonacci, prime, natural, squares and odd weights are used for feature extraction, then maximum possible values are 54, 77, 36, 204 and 64 respectively. The different weighting schemes considered for feature extraction with their range, maximum value and feature vector (fv) length is shown in table 3.1. Thus, by using fibonacci, prime, natural, squares and odd weights, the feature vector length gets reduced by 78.51%, 69.53%, 85.54%, 19.92% and 74.61% respectively with reference to binary weights. In figure 3.5, feature response maps generated by Rook, Bishop, Knight, Rook_Knight, Rook_Bishop and Knight_Bishop features using binary weights is shown.

$$hist_r = Hist(Rook) \tag{3.17}$$

$$hist_b = Hist(Bishop) \tag{3.18}$$

$$hist_k = Hist(Knight) \tag{3.19}$$

$$hist_{r\_k} = Hist(Rook\_Knight) \tag{3.20}$$

$$hist_{r\_b} = Hist(Rook\_Bishop) \tag{3.21}$$

Algorithm 3.1 Feature Extraction through CP

Input: An Input image (Img) of size N x N
Output: Feature vector (fv_{cp}) of size $\lceil$(N-4)/C$\rceil$ * $\lceil$(N-4)/C$\rceil$ * 6 * L

procedure CP(Img)
    Initialization: Rook, Bishop, Knight, Rook_Bishop, Rook_Knight, Knight_Bishop $\leftarrow \{\ \}$
    Load an input image
    for all a $\in$ range(1,N-4) **do**
        for all b $\in$ range(1,N-4) **do**
            Block = img(a:a+4,b:b+4)
            Assign the pixel values to Rook, Bishop and Knight in the 5 x 5 block.
            Calculate six features using the eqs.(3.3) to (3.9).
        end for
    end for
    Create six feature response maps from six features by reshaping them to (N-4) x (N-4).
    Each feature response map is partitioned into C x C non-overlapping blocks.
    Histograms are extracted block wise from all the feature response maps using eqs.(3.17) to (3.22).
    Concatenate all histograms to obtain feature vector fv_{cp} using eq.(3.23).
    return fv_{cp}
end procedure

$$hist_{k\_b} = Hist(Knight\_Bishop) \tag{3.22}$$

$$fv_{\text{cp}} = hist_r \cup hist_b \cup hist_k \cup hist_{r\_k} \cup hist_{r\_b} \cup hist_{k\_b} \tag{3.23}$$

Six feature response maps are created from the extracted six features. Research suggests that block wise feature extraction boosts up recognition accuracy [17]. Following the same, each feature response map is then partitioned into C x C non-overlapping blocks. Next, block wise histogram features are extracted from each of these six feature response maps, as shown in eqs. (3.17) to (3.22). Hist(.) corresponds to the histogram extraction function. hist_r, hist_b, hist_k, hist_{r_k}, hist_{r_b}, hist_{k_b} are the block wise histograms of Rook, Bishop, Knight, Rook_Knight, Rook_Bishop, Knight_Bishop respectively. $\cup$ represents concatenation operation and the final feature vector corresponding to CP (fv_{cp}) is formed by concatenating all the histograms obtained from each of the six feature response maps, as shown in eq.(3.23). The algorithm for CP is demonstrated in algorithm 3.1. For an input image of size N x N, the computational complexity of the proposed CP method is $O(N^2)$.

The value of 'L' in algorithm 3.1 depends on w_x used for feature extraction. After feature vector generation, for the purpose of classification, a multi-class SVM classifier with a linear kernel is employed for classifying the query images into various expressions.

3.3 Knight Tour Patterns

3.3.1 Knight's Tour

In a 'n x n' chess board, a Knight's tour is defined as the series of moves taken by the Knight for visiting every square exactly once [165]. As per chess game rules, a Knight can jump to any square in 'L' shape either by moving two squares vertically and one square horizontally (or) by moving two squares horizontally and one square vertically. In graph theory, finding the Knight tour is an example of the Hamiltonian path problem [166]. In general, there are two kinds of solutions to the Knight's tour problem namely the closed and the open knight tours [167]. If the last square visited by the Knight is reachable from the first square with a single Knight's move, then the Knight's tour is said to be closed. A Knight's tour in which every square is visited exactly once, but without being able to return to the first square in a single Knight's move is known as the open Knight's tour. In a 'n x n' chess board, the number of knight tours depend on the position at which the Knight starts. The variants of Knight tour problem involves the chess boards of different sizes rather than traditional 8 x 8 chess boards.

Inspired by the Knight tour problem in graph theory, novel local texture based feature descriptors named Knight Tour Patterns (kTP and KTP) have been proposed for facial feature extraction. Based on the Knight's moves, kTP and KTP are proposed for extracting facial features in 3 x 3 and 5 x 5 neighborhoods respectively. In a 3 x 3 neighborhood, a Knight can cover only the neighboring pixels and can never reach the center pixel (p_c), by starting from any index. Hence, the series of Knight moves on the neighboring pixels are considered for feature extraction through kTP. Indexes in a 3 x 3 block are shown in figure 3.6(a). Sample 3 x 3 block with names given to neighboring pixels and center pixel, is shown in 3.6(b). In a 3 x 3 block also, the pixels surrounding the pixel p_c are named as

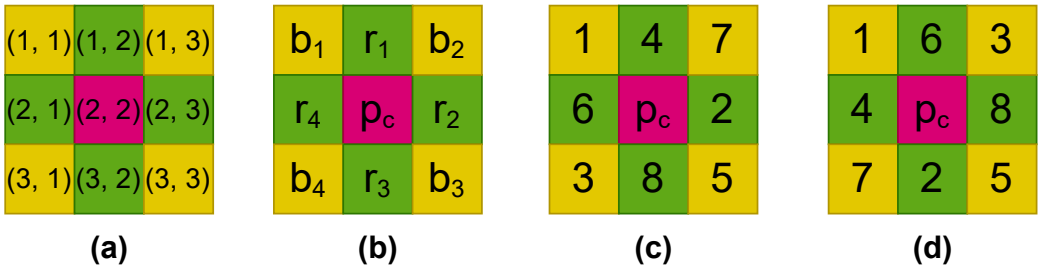


Figure 3.6: (a) Sample 3 x 3 block with indexes (b) Sample 3 x 3 block with names given to neighboring pixels and center pixel. Numbering is given for sequence of pixels considered for feature extraction through (c) kTP_1 (d) kTP_2

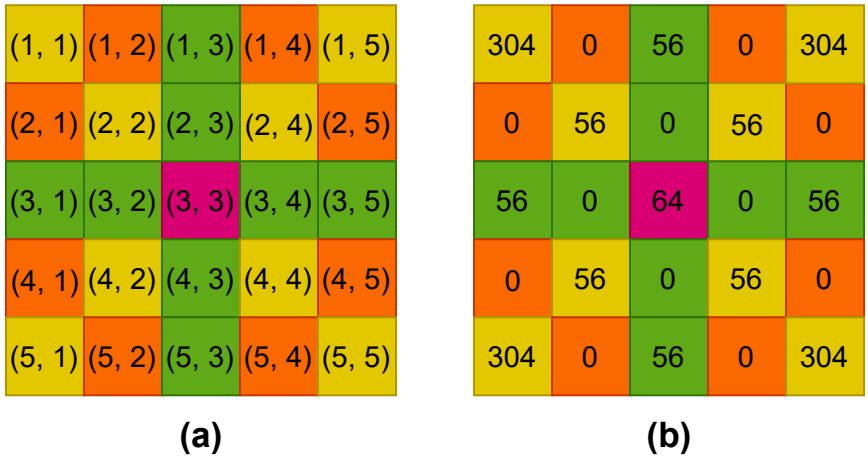


Figure 3.7: (a) Indexes in a 5 x 5 block (b) Number of possibilities of 5 x 5 Knight tour based on starting index

per chess game rules. The four-neighbors (r_1, r_2, r_3 and r_4) logically correspond to Rook positions and the diagonal neighbors (b_1, b_2, b_3 and b_4) correspond to the Bishop positions. Indexes in a 5 x 5 block are shown in figure 3.7(a). In a 5 x 5 neighborhood, there are many Knight tours possible based on the starting index, as shown in figure 3.7(b). If the Knight's initial move starts at index (1, 1), then 304 Knight tours are possible. Similarly, there are 56 and 64 Knight tours possible, if the initial Knight move starts at indexes (2, 2) and (3, 3) respectively. One among many such possibilities of Knight tour in a 5 x 5 neighborhood, where the initial move starting at index (1, 1) and ending at the index of pixel p_c (3, 3), has been chosen for feature extraction through KTP. The process of feature extraction through kTP is mentioned in subsection 3.3.2 and that of KTP is mentioned in subsection 3.3.3.

3.3.2 Knight Tour Pattern (kTP) Feature Extraction

kTP extracts two features namely kTP_1 and kTP_2 based on Knight moves in a 3 x 3 neighborhood. Starting at an index (1, 1) in a 3 x 3 neighborhood, a Knight can jump in 'L' shape either in horizontal or in vertical directions.

3.3.2.1 kTP_1 Feature Extraction

For feature extraction through kTP_1, starting from index (1, 1), initially, the Knight move in horizontal direction is considered. Generally, the neighboring pixels are compared with pixel p_c, for capturing discriminative information in a local neighborhood [46, 163]. Following the same methodology, in the proposed kTP method, the pixels visited based on the Knight moves are sequentially compared with pixel p_c, as shown in eq. (3.24) and in figure 3.6(c). The Knight moves considered for feature extraction through kTP_1 are as follows: ($b_1 \rightarrow r_2 \rightarrow b_4 \rightarrow r_1 \rightarrow b_3 \rightarrow r_4 \rightarrow b_2 \rightarrow r_3$). Upon comparison with the pixel p_c, an eight bit binary sequence is obtained which is then multiplied with the weight matrix (w_m), as shown in eq.(3.25). In image processing applications, binary weights are generally used for feature extraction. For kTP_1 method, apart from using binary weights, other weights such as fibonacci, prime, natural, squares and odd have been utilized for determining the optimal recognition accuracy. The equations corresponding to different weights have been shown in eqs.(3.26) to (3.32). The weights considered in eqs.(3.11) to (3.16) are in a decremental manner whereas the weights considered in eqs.(3.27) to (3.32) are in an incremental manner.

$$kTP_a = \{s(b_1, p_c), s(r_2, p_c), s(b_4, p_c), s(r_1, p_c), s(b_3, p_c), s(r_4, p_c), s(b_2, p_c), s(r_3, p_c)\} \tag{3.24}$$

$$kTP_1 = \sum (kTP_a. * w_m) \tag{3.25}$$

$$m = \{binary, fibonacci, prime, natural, squares, odd\} \tag{3.26}$$

$$binary = \begin{bmatrix} 1, & 2, & 4, & 8, & 16, & 32, & 64, & 128 \end{bmatrix} \tag{3.27}$$

$$fibonacci = \begin{bmatrix} 1, & 1, & 2, & 3, & 5, & 8, & 13, & 21 \end{bmatrix} \tag{3.28}$$

(a) (b) (c)

Figure 3.8: (a) Happy expression image from TFEID dataset. Feature response maps generated by (b) kTP_1 and (c) kTP_2 using binary weights.

$$prime = \Big[2, \quad 3, \quad 5, \quad 7, \quad 11, \quad 13, \quad 17, \quad 19\Big] \tag{3.29}$$

$$natural = \Big[1, \quad 2, \quad 3, \quad 4, \quad 5, \quad 6, \quad 7, \quad 8\Big] \tag{3.30}$$

$$squares = \Big[1, \quad 4, \quad 9, \quad 16, \quad 25, \quad 36, \quad 49, \quad 64\Big] \tag{3.31}$$

$$odd = \Big[1, \quad 3, \quad 5, \quad 7 \quad 9, \quad 11, \quad 13, \quad 15\Big] \tag{3.32}$$

3.3.2.2 kTP_2 Feature Extraction

For feature extraction through kTP_2, starting from index (1, 1), initially, the Knight move in vertical direction is considered, as shown in figure 3.6(d). The Knight moves considered for feature extraction through kTP_2 are as follows: ($b_1 \rightarrow r_3 \rightarrow b_2 \rightarrow r_4 \rightarrow b_3 \rightarrow r_1 \rightarrow b_4 \rightarrow r_2$). Upon comparison with the pixel p_c, an eight bit binary sequence is obtained which is then multiplied with the weight matrix (w_m), as shown in eqs.(3.33) and (3.34). For kTP_2 method also, apart from using binary weights, other weights such as fibonacci, prime, natural, squares and odd have been utilized for determining the optimal recognition accuracy.

$$kTP_b = \{s(b_1, p_c), s(r_3, p_c), s(b_2, p_c), s(r_4, p_c), s(b_3, p_c), s(r_1, p_c), s(b_4, p_c), s(r_2, p_c)\} \tag{3.33}$$

$$kTP_2 = \sum (kTP_b. * w_m) \tag{3.34}$$

Two feature response maps are created from the extracted two features. Then, each feature response map is partitioned into C x C non-overlapping blocks. Next, block wise

histogram features are extracted from each of these two feature response maps, as shown in eqs. (3.35) and (3.36). $hist_{kTP_1}$ and $hist_{kTP_2}$ are the block wise histograms of kTP_1 and kTP_2 respectively. The final feature vector corresponding to kTP (fv_{kTP}) is formed by concatenating all the histograms obtained from both of feature response maps, as shown in eq.(3.37). The algorithm for kTP is demonstrated in algorithm 3.2. For an input image of size N x N, the computational complexity of the proposed kTP method is $O(N^2)$. The value of 'L' in algorithm 3.2 depends on the weight vector w_m used for feature extraction. In figure 3.8, feature response maps generated by kTP_1 and kTP_2 using binary weights is shown.

Algorithm 3.2 Feature Extraction through kTP

Input: An Input image (Img) of size N x N
Output: Feature vector (fv_{kTP}) of size ⌈(N-2)/C⌉ * ⌈(N-2)/C⌉ * 2 * L

procedure KTP(Img)
  Initialization: kTP_1, $kTP_2 \leftarrow \{\ \}$
  Load an input image
  for all a ∈ range(1,N-2) **do**
    for all b ∈ range(1,N-2) **do**
      Block = img(a:a+2,b:b+2)
      Assign the pixel values to Rook and Bishop in the 3 x 3 block.
      Calculate kTP_1 features using eqs.(3.24) and (3.25)
      Calculate kTP_2 features using eqs.(3.33) and (3.34).
    end for
  end for
  Create two feature response maps from two features by reshaping them to (N-2) x (N-2).
  Each feature response map is partitioned into C x C non-overlapping blocks.
  Histograms are extracted block wise for both the feature response maps using eqs.(3.35) and (3.36).
  Concatenate both histograms to obtain feature vector fv_{kTP} using eq.(3.37).
  return fv_{kTP}
end procedure

$$hist_{kTP_1} = Hist(kTP_1) \tag{3.35}$$

$$hist_{kTP_2} = Hist(kTP_2) \tag{3.36}$$

$$fv_{kTP} = hist_{kTP_1} \cup hist_{kTP_2} \tag{3.37}$$

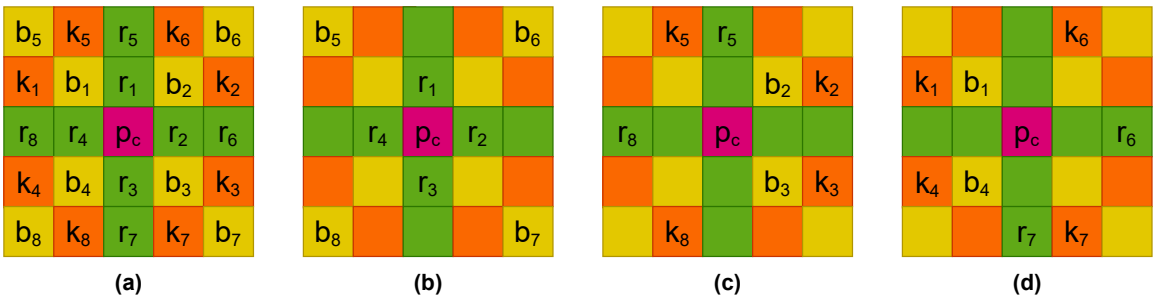


Figure 3.9: (a) Sample 5 x 5 block with numbering given to chessmen. Pixels considered for feature extraction through (b) KTP_1 (c) KTP_2 (d) KTP_3

3.3.3 Knight Tour Pattern (KTP) Feature Extraction

KTP extracts three features namely KTP_1, KTP_2 and KTP_3 in a 5 x 5 local neighborhood. Starting at an index (1, 1) in a 5 x 5 neighborhood, the Knight can cover all the twenty five pixel positions. For feature extraction through KTP, these twenty five pixel positions are logically divided into four groups. The first three groups contains eight pixels each, traversed by the Knight moves sequentially and the fourth group contains pixel p_c only. Initially, starting from index (1, 1), the Knight move in horizontal direction is considered. From there on, the pixels covered by the Knight moves are sequentially compared with the pixel p_c.

3.3.3.1 KTP_1 Feature Extraction

For feature extraction through KTP_1, the first group of pixels ($b_5 \rightarrow r_1 \rightarrow b_6 \rightarrow r_2 \rightarrow b_7 \rightarrow r_3 \rightarrow b_8 \rightarrow r_4$) are sequentially compared with the pixel p_c, as shown in eq.(3.38) and in figure 3.9(b). Upon comparison, the corresponding binary number thus obtained is multiplied with w_m, as shown in eq.(3.39). Thus, KTP_1 considers horizontal and vertical neighbors in the 3 x 3 neighborhood and diagonal neighbors in the 5 x 5 neighborhood.

$$KTP_c = \{s(b_5, p_c), s(r_1, p_c), s(b_6, p_c), s(r_2, p_c), s(b_7, p_c), s(r_3, p_c), s(b_8, p_c), s(r_4, p_c)\} \tag{3.38}$$

$$KTP_1 = \sum (KTP_c . * w_m) \tag{3.39}$$

3.3.3.2 KTP_2 Feature Extraction

For feature extraction through KTP_2, the second group of pixels are sequentially compared with the pixel p_c, which contains the next set of eight pixels traversed by the Knight moves ($r_5 \rightarrow k_2 \rightarrow b_3 \rightarrow k_8 \rightarrow r_8 \rightarrow k_5 \rightarrow b_2 \rightarrow k_3$) are considered and are compared with the pixel p_c, as shown in eq.(3.40) and in figure 3.9(c). Upon comparison, an eight bit binary sequence is obtained, which is then multiplied with w_m, as shown in eq.(3.41).

$$KTP_d = \{s(r_5, p_c), s(k_2, p_c), s(b_3, p_c), s(k_8, p_c), s(r_8, p_c), s(k_5, p_c), s(b_2, p_c), s(k_3, p_c)\} \tag{3.40}$$

$$KTP_2 = \sum (KTP_d. * w_m) \tag{3.41}$$

3.3.3.3 KTP_3 Feature Extraction

For feature extraction through KTP_3, the third group of pixels are sequentially compared with the pixel p_c, which contains the next set of eight pixels traversed by the Knight moves ($r_7 \rightarrow k_4 \rightarrow b_1 \rightarrow k_6 \rightarrow r_6 \rightarrow k_7 \rightarrow b_4 \rightarrow k_1$) are considered and are compared with the pixel p_c, as shown in eq.(3.42) and in figure 3.9(d). Upon comparison, an eight bit binary sequence is obtained, which is then multiplied with w_m, as shown in eq.(3.43).

$$KTP_e = \{s(r_7, p_c), s(k_4, p_c), s(b_1, p_c), s(k_6, p_c), s(r_6, p_c), s(k_7, p_c), s(b_4, p_c), s(k_1, p_c)\} \tag{3.42}$$

$$KTP_3 = \sum (KTP_e. * w_m) \tag{3.43}$$

Three feature response maps are created from the extracted three features. Then, each feature response map is partitioned into C x C non-overlapping blocks. Next, block wise histogram features are extracted from each of these three feature response maps, as shown in eqs.(3.44) to (3.46). $hist_{KTP_1}$, $hist_{KTP_2}$ and $hist_{KTP_3}$ are the block wise histograms of KTP_1, KTP_2 and KTP_3 respectively. The final feature vector corresponding to KTP (fv_{KTP}) is formed by concatenating all the histograms obtained from all of the three feature response maps, as shown in eq.(3.47). The algorithm for KTP is demonstrated in algorithm 3.3. For an input image of size N x N, the computational complexity of the proposed KTP method

(a) (b) (c) (d)

Figure 3.10: (a) Happy expression image from TFEID dataset. Feature response maps generated by (b) KTP_1 (c) KTP_2 and (d) KTP_3 using binary weights.

is $O(N^2)$. The value of 'L' in algorithm 3.3 depends on w_m used for feature extraction. In figure 3.10, feature response maps generated by KTP_1, KTP_2 and KTP_3 using binary weights is shown. After feature vector generation, for the purpose of classification, a multi-class SVM classifier with a linear kernel is employed for classifying the query images into various expressions.

Algorithm 3.3 Feature Extraction through KTP

Input: An Input image (Img) of size N x N
Output: Feature vector (fv_{KTP}) of size ⌈(N-4)/C⌉ * ⌈(N-4)/C⌉ * 3 * L

procedure KTP(Img)
  Initialization: KTP_1, KTP_2, $KTP_3 \leftarrow \{\ \}$
  Load an input image
  for all a ∈ range(1,N-4) **do**
    for all b ∈ range(1,N-4) **do**
      Block = img(a:a+4,b:b+4)
      Assign the pixel values to Rook, Bishop and Knight in the 5 x 5 block.
      Calculate three features using the eqs.(3.38) to (3.43).
    end for
  end for
  Create three feature response maps from three features by reshaping them to (N-4) x (N-4).
  Each feature response map is partitioned into C x C non-overlapping blocks.
  Histograms are extracted block wise for all the three feature response maps using eqs.(3.44) to (3.46).
  Concatenate both histograms to obtain feature vector fv_{KTP} using eq.(3.47).
  return fv_{KTP}
end procedure

$$hist_{KTP_1} = Hist(KTP_1) \tag{3.44}$$

$$hist_{KTP_2} = Hist(KTP_2) \tag{3.45}$$

Table 3.2: Feature vector length comparisons of proposed methods

Method	Binary	Fibonacci	Prime	Natural	Squares	Odd
CP	1536	330	468	222	1230	390
kTP	512	110	156	74	410	130
KTP	768	165	234	111	615	195

$$hist_{KTP_2} = Hist(KTP_2) \tag{3.46}$$

$$fv_{\text{KTP}} = hist_{KTP_1} \cup hist_{KTP_2} \cup hist_{KTP_3} \tag{3.47}$$

3.4 Results and Comparison Analysis

In this section, the feature vector length comparison of proposed feature descriptors with different weights, the experimental results and the comparison of proposed methods with the existing methods is reported. For experimental analysis, the block size (C) is empirically chosen as 8.

3.4.1 Feature Vectors comparison

For each of the proposed methods, the feature vector length comparison using different weights is shown in table 3.2. As CP generates six features, it's fv length is three times the length of kTP and two times the length of KTP.

3.4.2 Experiments for Six Expressions

The experiments for six expressions have been conducted on different 'in the lab' datasets. The proposed methods have been implemented with different weights and the results have been tabulated. In table 3.3, for each dataset, the recognition accuracy comparison of CP, in table 3.4, the recognition accuracy comparison of kTP and in table 3.5, the recognition accuracy comparison of KTP with different weights is shown. Among the proposed methods with different weights, kTP method with binary weights achieved an optimal recognition accuracy of 62.28% on JAFFE dataset. KTP method with odd, binary and natural weights achieved an optimal recognition accuracy of 87.55%, 92.86% and 89.44% on

Table 3.3: Recognition accuracy of CP with different weights for six expressions on different 'in the lab' datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	58.74	59.02	58.82	58.33	58.77	58.92
MUG	86.23	86.37	85.79	86.82	85.56	87.41
CK+	91.12	91.34	91.23	92.43	91.22	91.56
OULU	73.96	74.58	73.68	74.81	74.79	74.92
TFEID	92.92	94.17	94.17	94.58	94.25	95.08
KDEF	84.04	83.80	83.80	83.57	84.29	83.81
WSEFEP	86.67	86.67	86.67	87.22	87.22	87.78
ADFES	91.67	91.67	91.67	91.67	91.67	92.42

Table 3.4: Recognition accuracy of kTP with different weights for six expressions on different 'in the lab' datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	62.28	61.70	61.17	60.05	60.08	57.30
MUG	86.88	86.96	86.52	86	86.88	85.33
CK+	91.23	91.77	90.59	91.56	91.67	90.91
OULU	75.76	75.97	75.69	75	76.11	75.21
TFEID	92.92	95.08	94.67	95.50	94.67	95.92
KDEF	83.81	84.76	84.05	82.38	83.09	83.57
WSEFEP	87.78	87.78	87.78	88.33	88.33	88.33
ADFES	90.91	94.70	91.67	93.18	95.45	91.67

Table 3.5: Recognition accuracy of KTP with different weights for six expressions on different 'in the lab' datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	60.03	59.41	60	59.51	58.82	59.93
MUG	87.26	87.11	87.26	87.04	86.82	87.55
CK+	92.86	92.20	92.43	91.55	92.64	91.77
OULU	75.97	75.90	75.28	75.68	75.97	74.24
TFEID	93.75	95	94.92	94.50	94.08	94.50
KDEF	83.57	83.81	84.05	82.86	83.81	82.38
WSEFEP	87.78	87.78	87.78	89.44	88.33	88.33
ADFES	93.18	93.94	92.42	93.94	92.42	90.91

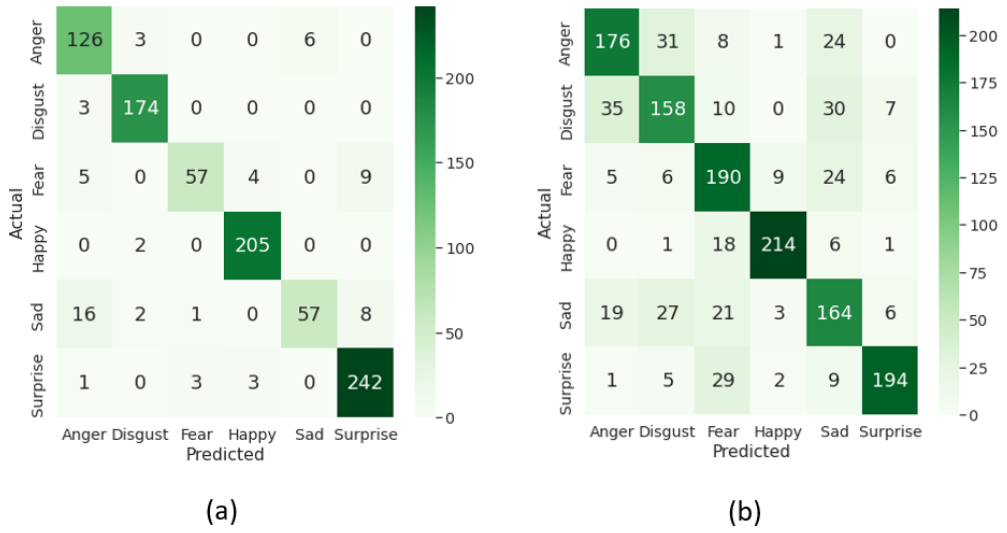


Figure 3.11: Confusion matrix for six expressions on (a) CK+ dataset using KTP method with binary weights (b) OULU dataset using kTP method with squares weights

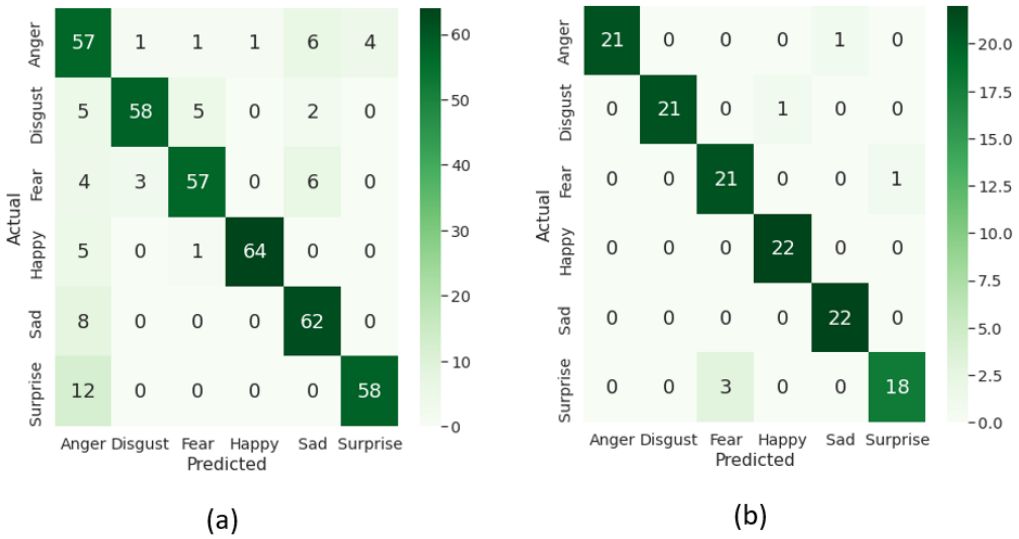


Figure 3.12: Confusion matrix for six expressions on (a) KDEF dataset using kTP method with fibonacci weights (b) ADFES dataset using kTP method with squares weights

MUG, CK+ and WSEFEP datasets respectively. The proposed kTP method with squares weights achieved an optimal recognition accuracy of 76.11% and 95.45% on OULU and ADFES datasets respectively. In case of TFEID and KDEF datasets, kTP method with odd and fibonacci weights achieved an optimal recognition accuracy of 95.92% and 84.76% respectively. The confusion matrix obtained using KTP method with binary weights for CK+ dataset is presented in figure 3.11(a) and for OULU dataset using kTP method with squares weights is presented in figure 3.11(b). The confusion matrix obtained using kTP method with fibonacci weights for KDEF dataset is presented in figure 3.12(a) and for ADFES dataset using kTP method with squares weights is presented in figure 3.12(b). In table 3.6, the comparison analysis of the proposed methods with the existing variants of binary patterns, implemented in our environment setup is shown. In table 3.7, the comparison analysis of the proposed methods with the existing methods is shown. In both tables 3.6 and 3.7, the proposed methods and their recognition accuracy has been highlighted in bold. In figure 3.13, the comparison analysis of proposed method with the existing variants of binary patterns on JAFFE and OULU datasets is shown. Among the proposed methods (CP, kTP and KTP), which ever method gave the highest recognition accuracy, that method is chosen as the proposed method for comparison with existing variants of binary patterns in figure 3.13. This notion is followed throughout the thesis.

The comparison analysis for JAFFE dataset with the existing variants of binary patterns is reported in the second column of table 3.6. From table 3.6, the proposed kTP method with binary weights outperformed the existing variants of binary patterns such as MSBP, LDSP, LDDSCP, RADAP and LBP + LNEP by 5.53%, 5.58%, 2.73%, 5.06% and 0.99% respectively. From table 3.7, the proposed kTP method also outperformed the existing methods such as ResNet50, LOOP and WLGC-HD by 2.84%, 3.56% and 1.58% respectively. The comparison analysis for MUG dataset with the existing variants of binary patterns is reported in the third column of table 3.6. From table 3.6, the proposed KTP method with odd weights outperformed the existing variants of binary patterns such as MSBP, LDSP, LDDSCP, RADAP and LBP + LNEP by 1.77%, 2.36%, 1.4%, 4.07% and 1.03% respectively. From table 3.7, the proposed KTP method also outperformed the existing methods such as ResNet50, DAGSVM and LOOP by 0.67%, 5.27% and 2.22% re-

Table 3.6: Comparison analysis with existing variants of binary patterns for six expressions on different 'in the lab' datasets

Method	JAFFE	MUG	CK+	OULU	TFEID	KDEF	WSEFEP	ADFES
LBP [46]	56.66	82.65	89.97	75.34	92.42	80.95	87.22	90.16
LDP [97]	52.77	82.87	90.84	72.43	93.67	80.95	86.67	90.91
LDN [31]	56.66	81.96	88.84	72.29	93.33	82.62	87.22	88.64
CSLBP [28]	53.28	83.94	90.68	71.53	94.25	81.67	84.44	87.12
LGC [84]	58.89	86.22	89.61	73.47	95.00	83.81	87.78	90.91
LDTP [32]	55.55	82.04	89.60	72.29	93.25	83.57	87.78	88.64
LDTeRP [33]	58.54	80.15	85.89	68.54	90.75	81.43	81.11	84.89
ALDP [35]	55.09	82.89	89.06	70.28	93.54	80.95	85.00	85.61
MSBP [88]	56.75	85.78	90.58	73.41	94.25	83.10	86.67	90.15
LDSP [34]	56.70	85.19	91.54	68.47	94.50	82.38	85.00	86.36
LDDSCP [99]	59.55	86.15	89.61	73.47	95.50	84.76	83.89	89.39
RADAP [17]	57.22	83.48	90.63	75.90	94.17	82.38	87.78	91.67
LBP + LNEP [134]	61.29	86.52	92.21	75.00	93	83.57	88.89	90.15
CP	**59.02**	**87.41**	**92.43**	**74.92**	**95.08**	**84.29**	**87.78**	**92.42**
kTP	**62.28**	**86.96**	**91.77**	**76.11**	**95.92**	**84.76**	**88.33**	**95.45**
kTP	**60.03**	**87.55**	**92.86**	**75.97**	**95**	**84.05**	**89.44**	**93.94**

Table 3.7: Comparison with existing methods for six expressions on different 'in the lab' datasets

Dataset	Method	Accuracy	Dataset	Method	Accuracy
JAFFE	IFRBC [86]	62.29	MUG	ResNet50 [17]	86.88
	ResNet50 [17]	59.44		DAGSVM [129]	82.28
	LOOP [11]	58.72		LOOP [11]	85.33
	WLGC-HD [57]	60.7		HiNet [108]	87.8
	CP	**59.02**		**CP**	**87.41**
	kTP	**62.28**		**kTP**	**86.96**
	KTP	**60.03**		**KTP**	**87.55**
CK+	ResNet50 [17]	89.32	OULU	ResNet50 [17]	73.1
	HiNet [108]	91.40		HiNet [108]	74.3
	WLGC-HD [57]	72.80		VGG16 [17]	73.4
	CP	**92.43**		**CP**	**74.92**
	kTP	**91.77**		**kTP**	**76.11**
	KTP	**92.86**		**KTP**	**75.97**
TFEID	DSNGE [85]	93.89	KDEF	IFRBC [86]	77.98
	DAMCNN [102]	93.65		ICVR [86]	76.31
	LIoP + HOG [123]	93.50		HOG [108]	82.17
	CP	**95.08**		**CP**	**84.29**
	kTP	**95.92**		**kTP**	**84.76**
	KTP	**95**		**KTP**	**84.05**
WSEFEP	LOOP [11]	87.78	ADFES	LOOP [11]	91.67
	CP	**87.78**		**CP**	**92.42**
	kTP	**88.33**		**kTP**	**95.45**
	KTP	**89.44**		**KTP**	**93.94**

spectively. In case of MUG dataset, HiNet method achieved 0.25% more than the proposed KTP method. The proposed methods are simple when compared to HiNet method which contains one million parameters.

The comparison analysis for CK+ dataset with the existing variants of binary patterns is reported in the fourth column of table 3.6. From table 3.6, the proposed KTP method with binary weights outperformed the existing variants of binary patterns such as MSBP, LDSP, LDDSCP, RADAP and LBP + LNEP by 2.28%, 1.32%, 3.25%, 2.23% and 0.65% respectively. From table 3.7, the proposed KTP method also outperformed the existing methods such as ResNet50, HiNet and WLGC-HD by 3.54%, 1.46% and 20.06% respectively. The comparison analysis for OULU dataset with the existing variants of binary patterns is reported in the fifth column of table 3.6. From table 3.6, the proposed kTP method with squares weights outperformed the existing variants of binary patterns such as MSBP, LDSP, LDDSCP, RADAP and LBP + LNEP by 2.7%, 7.64%, 2.64%, 0.21% and 1.11% respectively. From table 3.7, the proposed method kTP also outperformed the existing methods such as ResNet50, HiNet and VGG16 by 3.01%, 1.81% and 2.71% respectively.

The comparison analysis for TFEID dataset with the existing variants of binary patterns is reported in the sixth column of table 3.6. From table 3.6, the proposed kTP method with odd weights outperformed the existing variants of binary patterns such as MSBP, LDSP, LDDSCP, RADAP and LBP + LNEP by 1.67%, 1.42%, 0.42%, 1.75% and 2.92% respectively. From table 3.7, the proposed kTP method also outperformed the existing methods such as DSNGE, DAMCNN and LIoP + HOG by 2.03%, 2.27% and 2.42% respectively. The comparison analysis for KDEF dataset with the existing variants of binary patterns is reported in the seventh column of table 3.6. From table 3.6, the proposed kTP method with fibonacci weights outperformed the existing variants of binary patterns such as MSBP, LDSP, RADAP and LBP + LNEP by 1.66%, 2.38%, 2.38% and 1.19% respectively. From table 3.7, the proposed method kTP also outperformed the existing methods such as IFRBC, ICVR and HOG by 6.78%, 8.45% and 2.59% respectively.

The comparison analysis for WSEFEP dataset with the existing variants of binary patterns is reported in the eighth column of table 3.6. From table 3.6, the proposed KTP

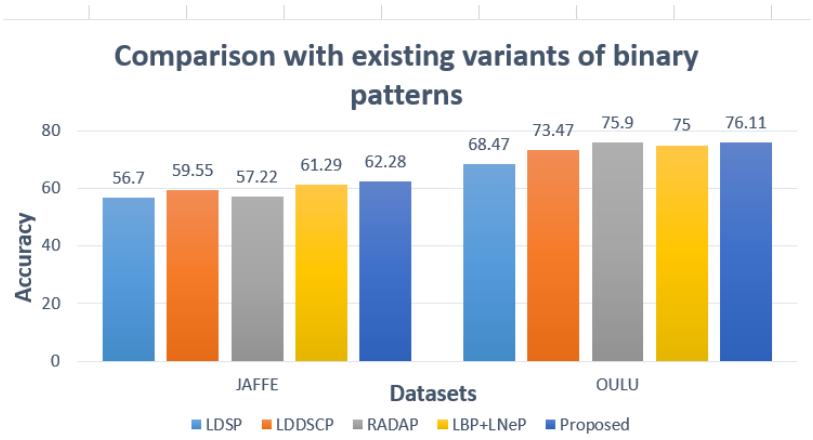


Figure 3.13: Comparison analysis of proposed method with existing variants of binary patterns for six expressions on JAFFE and OULU datasets

method with natural weights outperformed the existing variants of binary patterns such as MSBP, LDSP, LDDSCP, RADAP and LBP + LNEP by 2.77%, 4.44%, 5.55%, 1.66% and 0.55% respectively. From table 3.7, the proposed KTP method also outperformed the existing LOOP method by 1.66% respectively. The comparison analysis for ADFES dataset with the existing variants of binary patterns is reported in the ninth column of table 3.6. From table 3.6, the proposed kTP method with squares weights outperformed the existing variants of binary patterns such as MSBP, LDSP, LDDSCP, RADAP and LBP + LNEP by 5.3%, 9.09%, 6.06%, 3.78% and 5.3% respectively. From table 3.7, the proposed KTP method also outperformed the existing LOOP method by 3.78% respectively.

3.4.3 Experiments for Seven Expressions

The experiments for seven expressions have been conducted on different 'in the lab' datasets for CP, kTP and KTP methods. In addition to these datasets, the experiments have also been conducted for RAF and FERG datasets using kTP and KTP methods. The proposed methods have been implemented with different weights and the results have been tabulated. In table 3.8, for each dataset, the recognition accuracy comparison of CP, in table 3.9, the recognition accuracy comparison of kTP and in table 3.10, the recognition accuracy comparison of KTP with different weights is shown. Among the proposed methods with different weights, kTP method with natural weights achieved an optimal recognition accuracy of

Table 3.8: Recognition accuracy of CP with different weights for seven expressions on different 'in the lab' datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	58.50	58.93	59.97	57.68	58.05	59.04
MUG	81.26	82.79	81.80	81.90	82.35	82.41
CK+	85.94	86.17	85.94	86.13	86.22	86.15
OULU	72.08	72.86	72.44	72.08	72.56	72.44
TFEID	93.27	94.40	93.57	94.36	94.40	94.35
KDEF	81.43	82.05	81.63	82.25	82.05	82.04
WSEFEP	82.86	84.29	84.76	84.76	84.76	84.76
ADFES	92.21	92.86	92.21	91.56	92.21	92.21

60.99% on JAFFE dataset. kTP method with fibonacci weights achieved an optimal recognition accuracy of 83.11%, 86.19% and 94.81% on MUG, WSEFEP and ADFES datasets respectively. The proposed KTP method with natural and binary weights achieved an optimal recognition accuracy of 95% and 74.52% on TFEID and OULU datasets respectively. In case of CK+ and KDEF datasets, kTP method with squares weights and CP method with natural weights achieved an optimal recognition accuracy of 87.22% and 82.25% respectively. In case of RAF and FERG datasets, KTP method with odd and fibonacci weights achieved an optimal recognition accuracy of 77.35% and 99.99% respectively.

The confusion matrix obtained using kTP method with fibonacci weights for MUG dataset is presented in figure 3.14(a) and for TFEID dataset using KTP method with natural weights is presented in figure 3.14(b). The confusion matrix obtained using KTP method with odd weights for RAF dataset is presented in figure 3.15(a) and for FERG dataset using KTP method with fibonacci weights is presented in figure 3.15(b). In table 3.11, the comparison analysis of the proposed methods with the existing variants of binary patterns, implemented in our environment setup is shown. In table 3.12, the comparison analysis of the proposed methods with the existing methods is shown. The comparison analysis for RAF and FERG datasets with the existing methods is reported in table 3.13. In tables 3.11, 3.12 and 3.13, the proposed methods and their recognition accuracy has been highlighted in bold. In figure 3.16, the comparison analysis of proposed method with the existing variants of binary patterns on CK+ and WSEFEP datasets is shown.

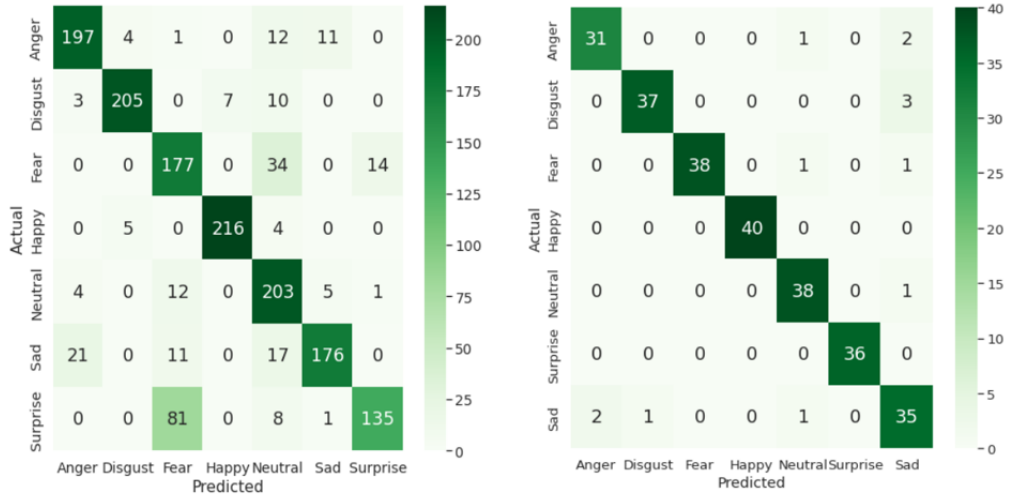
The comparison analysis for JAFFE dataset with the existing variants of binary patterns

Table 3.9: Recognition accuracy of kTP with different weights for seven expressions on different datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	58.52	59.09	58.97	60.99	58.41	60.04
MUG	81.02	83.11	82.34	82.28	82.28	81.52
CK+	87.03	86.88	86.70	85.65	87.22	85.99
OULU	73.81	74.11	74	72.80	73.45	73.39
TFEID	93.57	94.34	94.34	94.34	93.99	94.70
KDEF	80.82	81.84	81.84	80	81.02	80.61
WSEFEP	83.81	86.19	84.76	84.29	84.29	86.19
ADFES	90.26	94.81	92.21	92.86	93.51	93.51
RAF	73.11	73.76	74.05	72.06	74.45	73.08
FERG	98.67	99.84	98.77	98.99	99.06	98.31

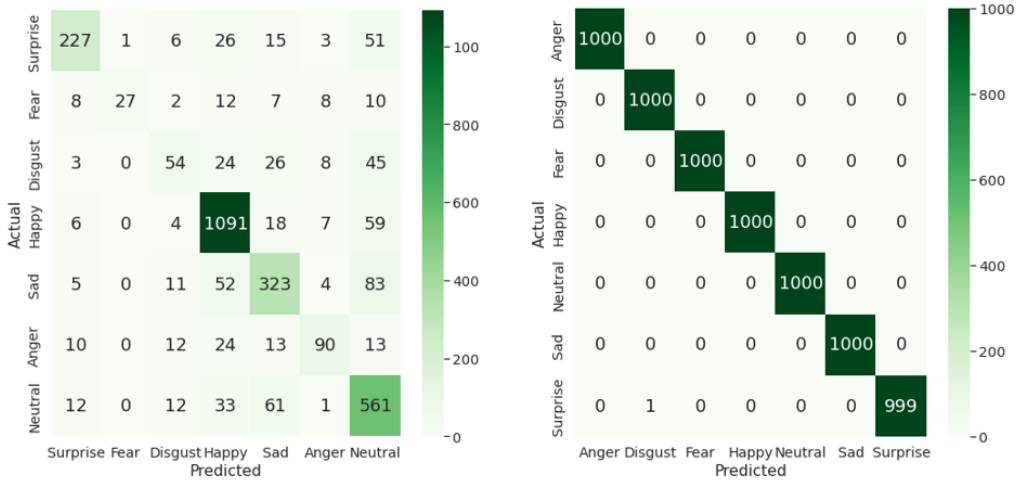
Table 3.10: Recognition accuracy of KTP with different weights for seven expressions on different datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	58.97	59.47	58.5	58.09	57.97	57.45
MUG	82.16	81.08	82.98	82.79	83.11	82.35
CK+	86.69	86.86	86.57	85.82	86.69	86.15
OULU	74.52	74.35	74.05	72.44	74.46	72.62
TFEID	93.99	94.34	93.93	95	93.93	94.28
KDEF	81.43	81.22	81.84	80.61	82.04	81.22
WSEFEP	85.24	84.76	84.29	85.71	85.71	85.24
ADFES	92.86	94.16	92.86	92.86	92.86	91.56
RAF	76.37	77.12	76.99	76.96	77.05	77.35
FERG	99.73	99.99	99.94	99.93	99.97	99.90



(a) (b)

Figure 3.14: Confusion matrix for seven expressions on (a) MUG dataset using kTP method with fibonacci weights (b) TFEID dataset using KTP method with natural weights



(a) (b)

Figure 3.15: Confusion matrix for seven expressions on (a) RAF dataset using KTP method odd weights (b) FERG dataset using KTP method with fibonacci weights

Table 3.11: Comparison analysis with existing variants of binary patterns for seven expressions on different 'in the lab' datasets

Method	JAFFE	MUG	CK+	OULU	TFEID	KDEF	WSEFEP	ADFES
LBP [46]	53.65	76.16	83.96	65.71	92.02	78.16	82.38	87.66
LDP [97]	52.10	78.70	84.80	69.16	86.20	80.61	80.95	90.26
LDN [31]	54.87	77.85	83.35	70.89	93.51	80.75	83.81	91.56
CSLBP [28]	52.40	79.37	85.51	57.98	89.44	80.20	79.44	86.36
LGC [84]	57.59	82.03	86.71	71.13	93.43	81.02	84.29	90.91
LDTP [32]	51.32	78.70	83.08	68.86	93.15	80.81	85.71	85.71
LDTeRP [33]	51.70	78.11	81.40	64.53	90.18	77.35	79.52	79.87
ALDP [35]	51.53	77.59	85.61	67.74	92.36	76.53	80.48	83.77
MSBP [88]	56.81	81.40	86.04	73.75	93.27	81.22	83.73	91.56
LDSP [34]	52.49	80.63	84.19	63.81	93.15	80.61	85.00	85.06
LDDSCP [99]	57.37	80.95	84.93	69.57	90.95	81.84	82.38	88.96
RADAP [17]	56.20	80.26	84.60	74.34	93.27	80.20	87.42	90.91
LBP + LNEP [134]	59.04	81.45	86.30	73.20	93.27	81.84	86.19	90.26
CP	**59.97**	**82.79**	**86.22**	**72.86**	**94.40**	**82.25**	**84.76**	**92.86**
kTP	**60.99**	**83.11**	**87.22**	**74.11**	**94.70**	**82.04**	**86.19**	**94.81**
kTP	**59.47**	**83.11**	**86.88**	**74.52**	**95**	**82.04**	**85.71**	**94.16**

Table 3.12: Comparison with existing methods for seven expressions on different 'in the lab' datasets

Dataset	Method	Accuracy	Dataset	Method	Accuracy
JAFFE	PCANet [117]	58.35	MUG	CBA [90]	78.57
	ResNet50 [17]	57.13		ResNet50 [17]	85.58
	LOOP [11]	59.67		LOOP [11]	79.68
	WLGC-HD [57]	58.2		HiNet [108]	87.2
	CP	**59.97**		**CP**	**82.79**
	kTP	**60.99**		**kTP**	**83.11**
	KTP	**59.47**		**KTP**	**83.11**
CK+	ResNet50 [17]	87.31	OULU	ResNet50 [17]	65.4
	HiNet [108]	88.6		HiNet [108]	72
	DLFS [107]	83.72		VGG19 [17]	70.5
	CP	**86.22**		**CP**	**72.86**
	kTP	**87.22**		**kTP**	**74.11**
	KTP	**86.88**		**KTP**	**74.52**
TFEID	DAMCNN [102]	93.36	KDEF	DLFS [107]	78.60
	Pyramid+SBDT [87]	93.38		PCANet [117]	69.59
	MSDV [132]	93.50		SAFL [108]	81.22
	CP	**94.40**		**CP**	**82.25**
	kTP	**94.40**		**kTP**	**82.04**
	KTP	**95**		**KTP**	**82.04**
WSEFEP	LOOP[11]	84.68	ADFES	AFM [103]	92.70
	CP	**84.76**		**CP**	**92.86**
	kTP	**86.19**		**kTP**	**94.81**
	KTP	**85.71**		**KTP**	**94.16**

Table 3.13: Comparison with existing methods for seven expressions on RAF and FERG datasets

Dataset	Method	Accuracy	Dataset	Method	Accuracy
RAF	DLPCNN [55]	74.20	FER	Deep Expr [56]	89.02
	ICID Fusion [101]	75.40		Ensemble Multi-Feature [17]	97
	Sadeghi et al. [93]	76.23		Adversarial NN [105]	98.2
	DCNN+RLPS [108]	72.84		Deep Emotion [106]	99.3
	IFSL [120]	76.9		LBP-AW [16]	96.7
	kTP	**74.45**		**kTP**	**99.84**
	KTP	**77.35**		**KTP**	**99.99**

is reported in the second column of table 3.11. From table 3.11, the proposed kTP method with natural weights outperformed the existing variants of binary patterns such as MSBP, LDSP, LDDSCP, RADAP and LBP + LNEP by 4.18%, 8.5%, 3.62%, 4.79% and 1.95% respectively. From table 3.12, the proposed kTP method also outperformed the existing methods such as PCANet, ResNet50, LOOP and WLGC-HD by 2.64%, 3.86%, 1.32% and 2.79% respectively. The comparison analysis for MUG dataset with the existing variants of binary patterns is reported in the third column of table 3.11. From table 3.11, the proposed kTP method with fibonacci weights weights outperformed the existing variants of binary patterns such as MSBP, LDSP, LDDSCP, RADAP and LBP + LNEP by 1.71%, 2.48%, 2.16%, 2.85% and 1.66% respectively. From table 3.12, the proposed kTP method also outperformed the existing methods such as CBA and LOOP by 4.54% and 3.43% respectively. In case of MUG dataset, HiNet and ResNet50 methods achieved 4.09% and 2.47% more than the proposed kTP method. The proposed methods are simple when compared to HiNet and ResNet50 methods which contains one million and thirty one million parameters respectively.

The comparison analysis for CK+ dataset with the existing variants of binary patterns is reported in the fourth column of table 3.11. From table 3.11, the proposed kTP method with squares weights outperformed the existing variants of binary patterns such as MSBP, LDSP, LDDSCP, RADAP and LBP + LNEP by 1.18%, 3.03%, 2.29%, 2.62% and 0.92% respectively. From table 3.12, the proposed kTP method outperformed the existing DLFS method by 3.5% respectively. Although, ResNet50 and HiNet methods achieved 0.09% and 1.38% better recognition accuracy than the proposed method, the proposed methods are simple and are easily implementable. The comparison analysis for OULU dataset with the existing variants of binary patterns is reported in the fifth column of table 3.11. From table 3.11, the proposed KTP method with binary weights outperformed the existing variants of binary patterns such as MSBP, LDSP, LDDSCP, RADAP and LBP + LNEP by 0.77%, 10.71%, 4.95%, 0.17% and 1.32% respectively. From table 3.12, the proposed method KTP also outperformed the existing methods such as ResNet50, HiNet and VGG19 by 9.12%, 2.52% and 4.02% respectively.

The comparison analysis for TFEID dataset with the existing variants of binary patterns

is reported in the sixth column of table 3.11. From table 3.11, the proposed KTP method with natural weights outperformed the existing variants of binary patterns such as MSBP, LDSP, LDDSCP, RADAP and LBP + LNEP by 1.73%, 1.85%, 4.05%, 1.73% and 1.73% respectively. From table 3.12, the proposed KTP method also outperformed the existing methods such as DAMCNN, Pyramid+SBDT and MSDV by 1.64%, 1.62% and 1.5% respectively. The comparison analysis for KDEF dataset with the existing variants of binary patterns is reported in the seventh column of table 3.11. From table 3.11, the proposed CP method with natural weights outperformed the existing variants of binary patterns such as MSBP, LDSP, LDDSCP, RADAP and LBP + LNEP by 1.03%, 1.64%, 0.41%, 2.05% and 0.41% respectively. From table 3.12, the proposed CP method also outperformed the existing methods such as DLFS, PCANet and SAFL by 3.65%, 12.66% and 1.03% respectively.

The comparison analysis for WSEFEP dataset with the existing variants of binary patterns is reported in the eighth column of table 3.11. From table 3.11, the proposed kTP method with fibonacci weights outperformed the existing variants of binary patterns such as MSBP, LDSP and LDDSCP by 2.46%, 1.19% and 3.81% respectively. From table 3.12, the proposed kTP method also outperformed the existing LOOP method by 1.51% respectively. The comparison analysis for ADFES dataset with the existing variants of binary patterns is reported in the ninth column of table 3.11. From table 3.11, the proposed kTP method with fibonacci weights outperformed the existing variants of binary patterns such as MSBP, LDSP, LDDSCP, RADAP and LBP + LNEP by 3.25%, 9.75%, 5.85%, 3.9% and 4.55% respectively. From table 3.12, the proposed kTP method also outperformed the existing LOOP method by 3.78% respectively. In case of RAF dataset, from table 3.13, the proposed KTP method with odd weights outperformed the existing methods such as DLPCNN, ICID Fusion, DCNN+RLPS and IFSL by 3.15%, 1.95%, 4.51% and 0.45% respectively. In case of FERG dataset, from table 3.13, the proposed KTP method with fibonacci weights outperformed the existing methods such as Deep Expr, Ensemble Multi-Feature, Adversarial NN, Deep Emotion and LBP-AW by 10.97%, 2.99%, 1.79%, 0.69% and 3.29% respectively.

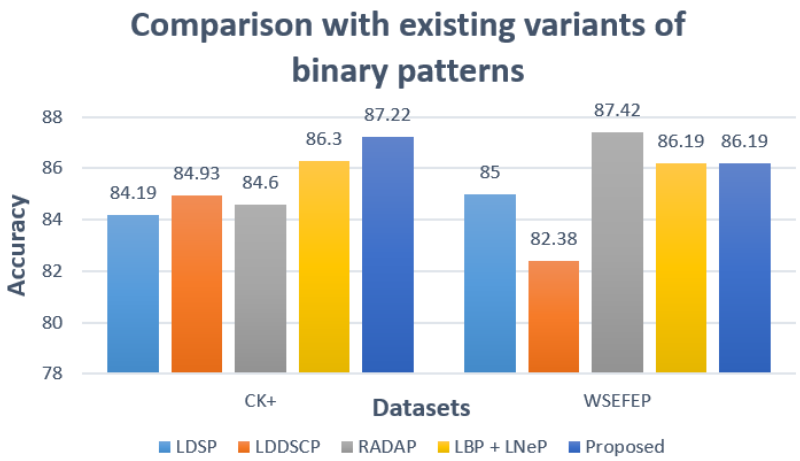


Figure 3.16: Comparison analysis of proposed method with existing variants of binary patterns for seven expressions on CK+ and WSEFEP datasets

3.4.4 Experiments for Eight Expressions

The experiments for eight expressions are performed on TFEID dataset. Apart from basic six plus neutral expressions, this dataset has one more expression named contempt. For the experimental evaluation, 336 images belonging to eight expressions have been considered. The proposed methods have been implemented with different weights and the results have been tabulated. In table 3.14, the recognition accuracy comparison of CP, in table 3.15, the recognition accuracy comparison of kTP and in table 3.16, the recognition accuracy comparison of KTP with different weights is shown. Among the proposed methods with different weights, CP method with odd weights achieved an optimal recognition accuracy of 91.69% for eight expressions on TFEID dataset. The existing variants of binary patterns have been implemented in our environment set up and correspondingly the recognition accuracy is reported in the second column of table 3.17. In table 3.17, the proposed methods and their recognition accuracy has been highlighted in bold. From table 3.17, the proposed CP method outperformed recent variants of binary patterns such as MSBP, LDSP, LDDSCP, RADAP and and LBP + LNEP by 2.15%, 5.01%, 4.03%, 1.13% and 0.88% respectively.

Table 3.14: Recognition accuracy of CP with different weights for eight and ten expressions on TFEID and ADFES datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
TFEID	90.48	90.44	91.07	90.55	90.75	91.69
ADFES	83.43	86.57	87.98	86.16	83.90	85.71

Table 3.15: Recognition accuracy of kTP with different weights for eight and ten expressions on TFEID and ADFES datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
TFEID	90.80	90.72	91.03	91.07	90.48	91.03
ADFES	87.07	89.49	88.59	85.25	87.07	86.62

3.4.5 Experiments for Ten Expressions

The experiments for ten expressions are performed on ADFES dataset. Apart from basic six plus neutral expressions, this dataset has three more expressions namely contempt, embarrass and pride. For the experimental evaluation, 215 images belonging to ten expressions have been considered. The proposed methods have been implemented with different weights and the results have been tabulated. In table 3.14, the recognition accuracy comparison of CP, in table 3.15, the recognition accuracy comparison of kTP and in table 3.16, the recognition accuracy comparison of KTP with different weights is shown. Among the proposed methods with different weights, kTP method with fibonacci weights achieved an optimal recognition accuracy of 89.49% for ten expressions on ADFES dataset. The existing variants of binary patterns have been implemented in our environment and correspondingly the recognition accuracy is reported in the third column of table 3.17. From table 3.16, the proposed kTP method outperformed recent variants of binary patterns such as MSBP, LDSP, LDDSCP, RADAP and and LBP + LNEP by 2.32%, 9.52%, 4.53%, 4.74% and 2.45% respectively.

Table 3.16: Recognition accuracy of KTP with different weights for eight and ten expressions on TFEID and ADFES datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
TFEID	91.31	90.75	91.48	89.89	90.72	89.57
ADFES	86.67	87.12	87.12	86.62	87.58	86.21

Table 3.17: Comparison analysis with the existing variants of binary patterns for eight and ten expressions on TFEID and ADFES datasets

Method	Eight Expressions	Ten Expressions
LBP [46]	90.95	83.54
LDP [97]	91.17	85.25
LDN [31]	88.06	85.20
CSLBP [28]	89.45	80.61
LGC [84]	89.95	85.20
LDTP [32]	86.29	82.68
LDTerP [33]	88.59	76.53
ALDP [35]	90.36	78.40
MSBP [88]	89.54	87.17
LDSP [34]	86.68	79.91
LDDSCP [99]	87.66	84.96
RADAP [17]	90.56	84.75
LBP + LNEP [134]	90.81	87.07
CP	**91.69**	**87.98**
kTP	**91.07**	**89.49**
KTP	**91.31**	**87.58**

3.5 Summary

In AFER systems, the main task is to accurately extract the features that best classify the expressions. In this regard, three local texture based feature descriptors namely CP, kTP and KTP have been presented in this Chapter. CP extracts six features based on the possible movements of Rook, Bishop and Knight in a 5 x 5 neighborhood. CP is presented with an intention to generate different feature codes for corner, edge and flat portions of an image. Inspired by the Knight tour problem in graph theory, kTP and KTP feature descriptors have been proposed which utilizes Knight moves for generating features by comparing neighboring pixels with the center pixel in a local neighborhood. The proposed CP, kTP and KTP methods have been implemented with different weights to determine the optimal recognition accuracy on standard FER datasets with respect to six, seven, eight and ten expressions. The experimental results demonstrated the efficiency of the proposed methods when compared to the existing methods.

Chapter 4

Modified Chess Patterns: Hand-crafted Feature Descriptors

Sign component has proven to be an efficient factor in developing feature descriptors [92]. Hence, in this Chapter, four texture based feature descriptors namely RMP, RCP, CSP and RCSP have been proposed for facial feature extraction by considering sign information. The main contributions of this Chapter are summarized as:

- RMP, a combination of RP and MP has been proposed with an intention to capture significant facial features in a local neighborhood by aiming to generate different feature codes for edge, corner and flat image regions.
- Local texture based feature descriptors namely RCP, CSP and their fusion RCSP are proposed and applied in a 5 x 5 neighborhood for extracting facial features by considering both the neighboring pixels relationship and the adjacent pixels relationship.

A brief description about the existing feature descriptors is presented in section 4.1. By drawing motivation from the existing descriptors such as Local Binary Pattern (LBP), LMeP and CP, a local texture based feature descriptor named RMP has been proposed for the purpose of facial feature extraction in section 4.2. Inspired by CP, Local Gradient Coding (LGC) and it's variants, feature descriptors such as RCP, CSP and RCSP have been proposed and are discussed in detail in sections 4.3, 4.4 and 4.5 respectively. In section

4.6, the experimental results corresponding to RMP, RCP, CSP and RCSP with different weights have been presented and analyzed. In section 4.7, the contributions in this Chapter have been summarized.

4.1 Preliminaries

A brief summary about existing feature descriptors such as LBP, LMeP, LGC and it's variants are explained in this section, as the proposed feature descriptors are developed based on these feature descriptors.

4.1.1 Local Binary Pattern (LBP)

LBP [46] is the most widely used texture based feature descriptor for capturing the pixel's intensity variations in a local neighborhood (3 x 3 region). The gray levels of eight neighboring pixels p_k (k=0,1,2,....,7) are compared with the gray level of the center pixel (p_c). At a pixel location (y, z), the LBP code is computed for the pixel p_c as given in eq.(4.1) as :

$$LBP(y,z) = \sum_{k=0}^{7} s(p_k, p_c) * 2^k \tag{4.1}$$

where, p_k represent the gray level values of neighboring pixels (k=0,1,....,7) and and p_c represent the gray value of center pixel.

4.1.2 Local Mesh Pattern (LMeP)

LMeP [168] is also a local texture based feature descriptor, that takes into consideration, the multi-distance information of the neighboring pixels in a local neighborhood. LMeP generates three feature codes, corresponding to multi-distance information (d $\in$ [1,3]). From a reference pixel, $LMeP_{p,1}$ is captured by comparing the subsequent adjacent pixels at a distance (d = 1), $LMeP_{p,2}$ is captured by comparing the subsequent adjacent pixels at a distance (d = 2) and $LMeP_{p,3}$ is captured by comparing the subsequent adjacent pixels at a distance (d = 3). Thus, corresponding to three distances considered, three feature codes are

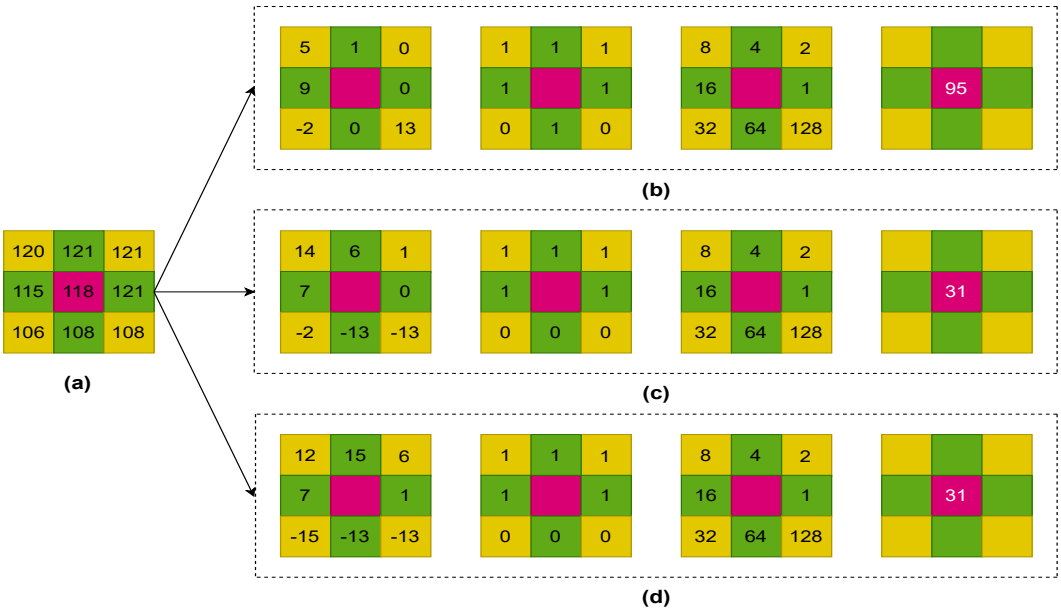


Figure 4.1: Example of feature extraction using LMeP (a) A sample 3 x 3 image patch (b) $LMeP_1$ (c) $LMeP_2$ and (d) $LMeP_3$

generated, which are finally concatenated to form final LMeP. The process of calculating LMeP code for a sample 3 x 3 image patch is shown in figure 4.1. At a given pixel location (y, z), LMeP is computed for the pixel p_c as represented in eq.(4.2) and eq.(4.3).

$$LMeP_{p,d}(y,z) = \sum_{i=0}^{p-1} s(x_{\tau_{i,d,p}}, x_i) \tag{4.2}$$

$$\tau_{i,d,p} = \begin{cases} \tau^1_{i,d,p}, & \text{if } \tau^1_{i,d,p} - 1 < p - 1 \\ \tau^2_{i,d,p}, & otherwise \end{cases} \tag{4.3}$$

where, $\tau^1_{i,d,p} = i + d$, $\tau^2_{i,d,p} = i + d - p$ and $d \in [1, 3]$ and p is the number of neighbors surrounding the pixel p_c. In LMeP, as binary weights are used for feature extraction, huge computational resources are required. Although, LMeP extracts better edge information than LBP, it achieves better recognition rate at the cost of high computational resources.

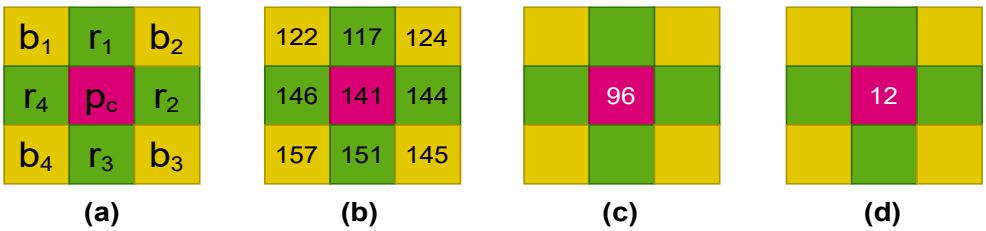


Figure 4.2: Feature extraction through LGC and LGC-HD (a) Sample mask for 3 x 3 operators (b) A sample 3 x 3 image patch (c) Computed feature code using LGC (d) Computed feature code using LGC-HD

4.1.3 Local Gradient Coding (LGC)

Tong et al. [84] proposed LGC for facial feature extraction. LGC extracts texture features in a 3 x 3 neighborhood. LGC operator encodes the gradient information in horizontal, vertical and diagonal directions to generate an eight bit binary number. The binary number thus formed is then converted into a decimal number, which is replaced in the place of pixel p_c. This process is repeated throughout the image, and all the histogram features are concatenated block wise to form a final feature vector. This LGC encoding captures consistent expression specific texture features in all possible directions. The coding formula for feature extraction through LGC operator is shown in eq.(4.4). As 3 x 3 mask is considered for LGC, usually radius (rd = 1) and number of neighbors (p = 8). In figure 4.2(a-c), feature extraction of LGC for a sample 3 x 3 numerical example is shown. To the LGC operator, some extensions are also proposed, which are Local Gradient Coding- based on Horizontal and Diagonal prior principle (LGC-HD) operator, LGC-FN operator (LGC based on a 5 × 5 neighborhood horizontal and diagonal gradient prior principle) and LGC-AD operator.

$$\begin{aligned} LGC_p^{rd} = s(b_1, b_2) * 2^7 + s(r_4, r_2) * 2^6 + s(b_4, b_3) * 2^5 + s(b_1, b_4) * 2^4 \\ + s(r_1, r_3) * 2^3 + s(b_2, b_3) * 2^2 + s(b_1, b_3) * 2^1 + s(b_2, b_4) * 2^0 \end{aligned} \tag{4.4}$$

4.1.3.1 Local Gradient Coding-Horizontal Diagonal (LGC-HD)

LGC-HD is also proposed by Tong et al. [84], further optimizes the LGC operator and decreases the characteristic feature vector length by considering the gradient information in horizontal and diagonal directions only. The coding formula for feature extraction through

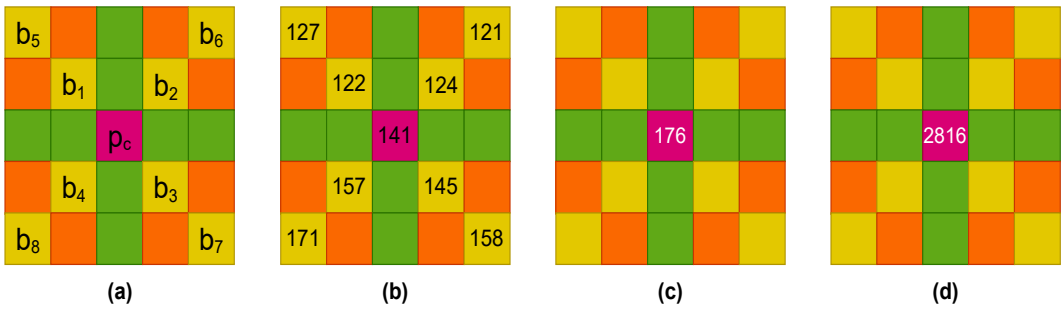


Figure 4.3: Feature extraction through LGC-FN and LGC-AD (a) Sample mask considered for 5 x 5 operators (b) A sample 5 x 5 image patch for LGC-FN (c) Computed feature code using LGC-FN (d) Computed feature code using LGC-AD

LGC-HD operator is shown in eq.(4.5). In figure 4.2(b,d), feature extraction of LGC-HD for a sample 3 x 3 numerical example is shown. As 3 x 3 mask is considered for LGC-HD, usually rd = 1 and p = 6.

$$LGC{-}HD_p^{rd} = s(b_1, b_2){*}2^4{+}s(r_4, r_2){*}2^3{+}s(b_4, b_3){*}2^2{+}s(b_1, b_3){*}2^1{+}s(b_2, b_4){*}2^0 \quad (4.5)$$

4.1.3.2 LGC-FN

LGC-FN operator [91, 169] expands LGC by considering 5 x 5 neighborhood size. LGC-FN computes feature codes in three directions namely in horizontal and along two diagonal directions. The coding formula for feature extraction through LGC-FN operator is shown in eq.(4.6). A sample mask considered for 5 x 5 operators is shown in figure 4.3 (a). In figure 4.3 (b), a sample 5 x 5 image patch is shown. In figure 4.3 (c), the computed feature code using LGC-FN is shown.

$$\begin{aligned} LGC - FN = {} & s(b_5, b_6) * 2^7 + s(b_1, b_2) * 2^6 + s(b_4, b_3) * 2^5 + s(b_8, b_7) * 2^4 \\ & + s(b_5, b_7) * 2^3 + s(b_1, b_3) * 2^2 + s(b_6, b_8) * 2^1 + s(b_2, b_4) * 2^0 \end{aligned} \quad (4.6)$$

4.1.3.3 LGC-AD

LGC-AD operator [91, 169] computes feature codes in four directions namely in horizontal, vertical and along two diagonal directions. Thus, LGC-AD operator is an extension of LGC-FN operator by additionally considering vertical gradient information. The coding

formula for feature extraction through LGC-AD operator is shown in eq.(4.7). In figure 4.3 (d), the computed feature code using LGC-AD is shown.

$$\begin{aligned} LGC-AD = {} & s(b_5,b_6)*2^{11}+s(b_1,b_2)*2^{10}+s(b_4,b_3)*2^9+s(b_8,b_7)*2^8 \\ & +s(b_5,b_8)*2^7+s(b_1,b_4)*2^6+s(b_2,b_3)*2^5+s(b_6,b_7)*2^4 \\ & +s(b_5,b_7)*2^3+s(b_1,b_3)*2^2+s(b_6,b_8)*2^1+s(b_2,b_4)*2^0 \end{aligned} \tag{4.7}$$

4.1.3.4 Limitations of existing descriptors

The existing feature descriptors have some limitations as follows:

- CP generates six feature codes, so it's feature vector (fv) length is six times the fv length of LBP. Also, it takes more computation time than traditional LBP operator.
- LGC method extracts features in a 3 x 3 neighborhood using three groups of horizontal pixels and three groups of vertical pixels, but in diagonal direction, only two groups of pixels are considered. As a result, the gradient information in the diagonal directions is not completely captured, which negatively impacts the recognition accuracy [91].
- In LGC-HD operator, fv length is reduced when compared to LGC, as it does not take into consideration the gradient information computed in the vertical direction.
- LGC-FN operator does not consider the gradient information in vertical information and also the characteristics of center pixel information is neglected.
- LGC-AD operator generates a fv length of 4096, which is very huge when compared to traditional LBP operator.
- Most of the existing edge based methods generate unstable patterns in the smoother regions of an image. Also, some of the existing variants of binary patterns generate the same feature codes for different image portions.

So, by considering all these information, new feature descriptors are developed, which are discussed in detail in the next sections.

4.2 Radial Mesh Pattern (RMP) Feature Extraction

By drawing motivation from the existing descriptors such as LBP, LMeP and CP, a local texture based feature descriptor named RMP is proposed for the purpose of facial feature extraction. RMP is different from the existing CP. The similarities and dissimlarities with existing descriptors are as follows:

- Both CP and RMP considers the movements of Rook, Bishop and Knight for feature extraction in a 5 x 5 neighborhood.
- LBP generates one feature code, LMeP generates three feature codes, whereas, CP generates six feature codes, each corresponding to Rook, Bishop, Knight, Rook_Bishop, Rook_Knight and Knight_Bishop. Whereas, RMP generates only two feature codes, each corresponding to Radial Pattern (RP) and Mesh Pattern (MP).
- For calculating RP, the concept of LBP is used. LBP computes feature code within a 3 x 3 local region only. Whereas, RP takes into consideration both 3 x 3 and 5 x 5 neighborhoods, for calculating feature codes. Thus, multi-level information is captured with RP.
- For calculating MP, the concept of LMeP is used. The computational cost of LMeP is high as it generates three feature codes. So, we only consider one LMeP pattern (d = 2), while designing the MP.
- Using CP, chessmen are numbered only in clockwise direction. Using LBP and LMeP, anti-clockwise direction is considered in the process of feature extraction. Whereas, RMP considers numbering in both anti-clockwise (for RP) as well as clockwise directions (for MP).
- LBP and LMeP uses only binary weights for feature extraction. Apart from binary weights, to further reduce the feature vector length of RMP, other weights such as fibonacci, prime, natural, squares and odd weights have been utilized.

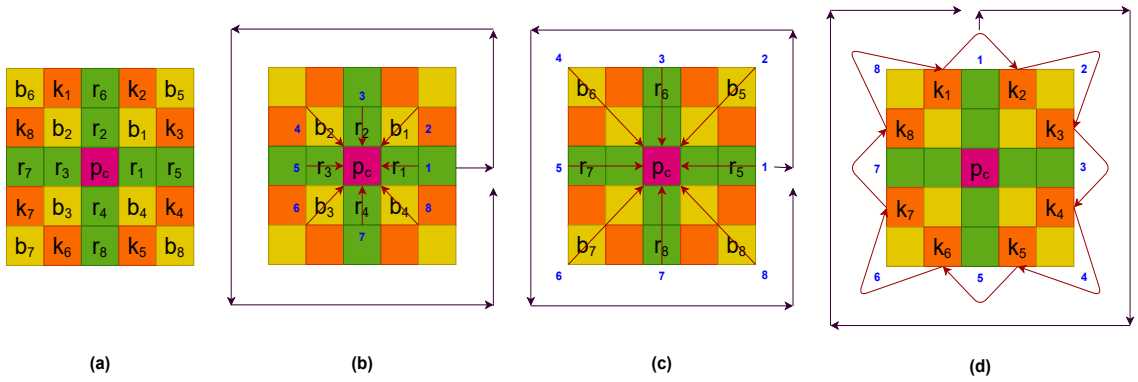


Figure 4.4: Process of feature extraction through RMP (a) Numbering given to chessmen in the 5 x 5 neighborhood as per RMP (b) Process of feature extraction through RP_1 (c) Process of feature extraction through RP_2 (d) Process of feature extraction through MP.

In image processing applications, selecting the neighborhood size is the key in the designing phase of hand-crafted feature descriptors. In general, if more pixels are used in designing a kernel, the more accurate is the classification. However, choosing a higher neighborhood size leads to increased computation time during the thresholding process. In this Chapter, for accommodating large inter-class distinctions and low intra-class variations, 5 x 5 neighborhood is adopted to explore wider information in a local neighborhood, which allows utilizing both radii (2) and angles (8) in designing the proposed RMP feature descriptor. Also, multi distance information is captured in a 5 x 5 neighborhood, as like the $LMeP_{p,2}$. In figure 4.4 (a), for feature extraction through RMP, numbering is assigned to the possible positions where Rook, Bishop and Knight could be placed. The numbering order for Rook (r) and Bishop (b) follows anti-clockwise direction and the numbering order for Knight (k) follows clockwise direction. Thus, RMP utilizes both anti-clockwise (for Rook and Bishop) and clockwise directions (for Knight) in the process of feature extraction. RMP is a combination of both RP and MP. The process of feature extraction through RP is explained in section 4.2.1 and through MP is explained in section 4.2.2.

4.2.1 Radial Pattern (RP)

Feature extraction through RP is based on the positions of Rook and Bishop in the entire 5 x 5 neighborhood. Initially, with reference to the pixel p_c, the possible positions of Rook and Bishop are numbered in the 3 x 3 neighborhood followed by 5 x 5 neighborhood. The

eight neighboring pixels in the 3 x 3 neighborhood (rd = 1) are compared with the pixel p_c. If the corresponding neighbor is greater than the pixel p_c, then the corresponding bit is encoded as one, else it is encoded as zero. As a result, a sequence of eight binary numbers is obtained, which is named as RP_1. The process of feature extraction through RP_1 is same as the methodology of LBP [46]. The only difference is that, in RP_1, the eight neighbors surrounding the pixel p_c are named after the possible positions of Rook and Bishop. The corresponding equation for calculating RP_1 is shown in eq.(4.8). The process of feature extraction through RP_1 is shown in figure 4.4 (b).

$$RP_1 = \{s(r_1, p_c), s(b_1, p_c), s(r_2, p_c), s(b_2, p_c), s(r_3, p_c), s(b_3, p_c), s(r_4, p_c), s(b_4, p_c)\} \tag{4.8}$$

Next, with reference to the pixel p_c, the eight neighbors in the 5 x 5 neighborhood (rd = 2) corresponding to angles ($0°$, $45°$, $90°$, $135°$, $180°$, $225°$, $270°$, $315°$) are compared with the pixel p_c. If the corresponding neighbor is greater than the p_c, then the corresponding bit is encoded as one, else it is encoded as zero. From this operation, a sequence of eight binary numbers is obtained, which is named as RP_2. In RP_2 also, the eight neighbors surrounding the pixel p_c are named after the possible positions of Rook and Bishop. The corresponding equation for calculating RP_2 is shown in the eq.(4.9). The process of feature extraction through RP_2 is shown in figure 4.4 (c). The final RP is obtained by performing logical AND operation between the obtained patterns RP_1 and RP_2. The resultant binary number thus formed is multiplied by w_m. The corresponding equation for calculating RP is shown in eq.(4.10). Usually, binary weights are used in the calculation of feature vectors. In order to minimize feature vector length and to maximize recognition accuracy, other weights such as fibonacci [164], prime, natural, squares and odd have been utilized for facial feature extraction. The corresponding equations for different weights are shown in eq.(3.26) to eq.(3.32).

$$RP_2 = \{s(r_5, p_c), s(b_5, p_c), s(r_6, p_c), s(b_6, p_c), s(r_7, p_c), s(b_7, p_c), s(r_8, p_c), s(b_8, p_c)\} \tag{4.9}$$

$$RP = \sum((RP_1 \wedge RP_2). * w_m) \tag{4.10}$$

4.2.2 Mesh Pattern (MP)

The basis for designing MP is derived from LMeP. LMeP generates three feature codes by encoding the multi-distance relationships between the neighboring pixels, surrounding the pixel p_c. But, for MP, the concept of $LMeP_{p,2}$ is only adopted. Based on the Knight positions in the entire 5 x 5 neighborhood, MP is obtained. Based on the numbering assigned, the pixels present in the possible Knight positions are compared sequentially. If the initial pixel corresponding to Knight position is greater than the subsequent pixel, then the corresponding bit is encoded as one, else it is encoded as zero. The eight bit sequence thus obtained corresponds to the MP. The corresponding equations for calculating MP are shown in eqs.(4.11) and (4.12). The resultant binary number thus formed is multiplied by w_m. The process of feature extraction through MP is shown in figure 4.4(d).

$$MP = \{s(k_1, k_2), s(k_2, k_3), s(k_3, k_4), s(k_4, k_5), s(k_5, k_6), s(k_6, k_7), s(k_7, k_8), s(k_8, k_1)\} \tag{4.11}$$

$$MP = \sum(MP. * w_m) \tag{4.12}$$

4.2.3 Feature Level Fusion

RMP is obtained by horizontally concatenating both RP and MP. The corresponding equation for calculating RMP is shown in eq.(4.13). The process of feature extraction through RMP is demonstrated in figure 4.4. Aiming to develop feature descriptors with low dimensionality and reduced computation time, different weighting schemes such as fibonacci, prime, natural, squares, odd and even weights have been utilized for the purpose of feature extraction, which are shown in eqs.(3.26) to (3.32). Thus, w_m can take any one of the weight values among binary, fibonacci, prime, natural, squares and odd. So, corresponding to six different weights (binary, fibonacci, prime, natural, squares and odd), six different variants of RMP namely RMP_binary, RMP_fibonacci, RMP_prime, RMP_natural, RMP_squares and RMP_odd have been proposed. For example, RMP_prime method uses prime weights (shown in eq.(3.29)) and RMP_odd method uses odd weights (shown in eq. (3.32)) for

(a) (b) (c)

Figure 4.5: (a) Happy expression image from TFEID dataset. Feature response maps generated by (b) RP and (c) MP using binary weights.

extracting facial features.

$$RMP = RP \cup MP \tag{4.13}$$

The feature response maps generated by RP and MP methods using binary weights is shown in figure 4.5. The feature codes generated by existing methods such as LBP, LDP, LDN, PTP, LDDSCP, LOOP and the proposed method, RMP for different image portions such as corner, edge and flat are shown in figure 4.6. From figure 4.6, the proposed RMP could generate different feature codes for different image portions. The algorithm for feature extraction through RMP is mentioned in algorithm 4.1. The value of 'L' in algorithm 4.1 depends on the weight matrix used in the process of feature extraction. For an input image of size N x N, the computational complexity of the proposed RMP method is O(N^2).

$$hist_{rp} = Hist(rp) \tag{4.14}$$

$$hist_{mp} = Hist(mp) \tag{4.15}$$

$$fv_{\text{rmp}} = hist_{rp} \cup hist_{mp} \tag{4.16}$$

$$Size(fv_{\text{rmp}}) = \lceil (N-4)/C \rceil * \lceil (N-4)/C \rceil * 2 * L \tag{4.17}$$

Two feature response maps are created from the extracted two features. Then, each feature response map is partitioned into C x C non-overlapping blocks. Next, block wise histogram features are extracted from each of these two feature response maps, as shown in eqs. (4.14) and (4.15). $hist_{rp}$ and $hist_{mp}$ are the block wise histograms of RP and MP respectively. The final feature vector (fv_{rmp}) is formed by concatenating all the histograms obtained from each of the two feature response maps, as shown in eq.(4.16). The equation for calculating the size of feature vector is shown in eq.(4.17). The value of 'L' in eq. (4.17)

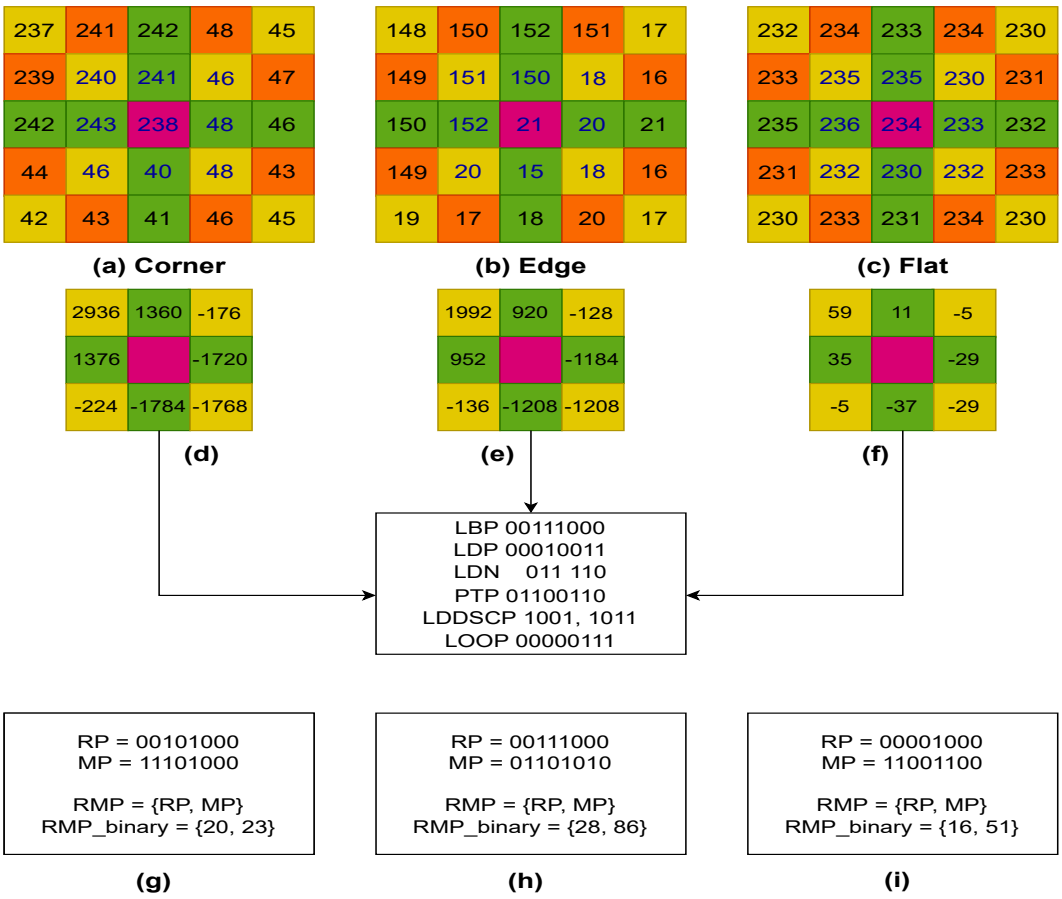


Figure 4.6: Example for drawbacks of existing feature descriptors. 5 x 5 sample image portions corresponding to (a) corner (b) edge (c) flat image regions. (d-f) Kirsch responses for the regions (a-c). (g) Same feature codes are generated by existing methods for different image regions (a-c). (g-i) RMP generating different codes for different edge patterns (a-c). In (g-i), the generated feature codes with binary weights is also shown

depends on the w_m used in the process of feature extraction. The value of 'C' corresponds to the block size.

4.3 Radial Cross Pattern (RCP) Feature Extraction

RCP considers multi-radial and multi-orientation information and extracts two features namely RCP_1 and RCP_2 by comparing the neighboring pixels with the current pixel in horizontal, vertical and diagonal directions.

Algorithm 4.1 Feature Extraction through RMP

Input: An Input image (Img) of size N x N
Output: Feature vector (fv_{rmp}) of size $\lceil$(N-4)/C$\rceil$ * $\lceil$(N-4)/C$\rceil$ * 2 * L

procedure RMP(Img)
  Initialization: RP_1, RP_2, RP, MP $\leftarrow$ { }
  Load an input image (Img)
  for all a $\in$ range(1,N-4) **do**
    for all b $\in$ range(1,N-4) **do**
      Block = img(a:a+4,b:b+4)
      Assign the pixel values to Rook, Bishop and Knight in the 5 x 5 block.
      Calculate RP using eqs.(4.9) to (4.11)
      Calculate MP using eqs.(4.11) and (4.12)
    end for
  end for
  Create two feature response maps obtained from RP and MP by reshaping them to (N-4) x (N-4).
  Each feature response map is partitioned into C x C non-overlapping blocks.
  Histograms are extracted block wise for both the feature response maps using eqs.(4.14) and (4.15).
  Concatenate all the histograms to obtain feature vector fv_{rmp} using eq.(4.16).
  return fv_{rmp}
end procedure

4.3.1 RCP_1

RCP_1 contains a set of eight pixels, which includes four pixels corresponding to Rook, considered from the 3 x 3 neighborhood (rd = 1) and the remaining four pixels corresponding to Bishop, considered from the 5 x 5 neighborhood (rd = 2). The numbering system for Rook and Bishop follows anti-clockwise direction as per Moore's neighborhood [17], as shown in figure 4.7 (a). The pixel positions corresponding to RCP_1 are shown in figure 4.7(b). Thus, considering pixels in this manner enables in better capturing the expression specific texture information in eight directions (0°, 90°, 180°, 270°, 45°, 135°, 225°, 315°) respectively. The pixel intensities present in these eight positions ($r_{1,2,3,4}$, $b_{5,6,7,8}$) are compared with the intensity of pixel p_c. Upon comparison, if the obtained result is positive, then the corresponding bit is encoded as one, else it encoded as zero. Thus, for eight pixel positions, eight corresponding values (either 0 or 1) are obtained, which are then concatenated to form an eight bit binary number, which is subsequently multiplied with w_m. The

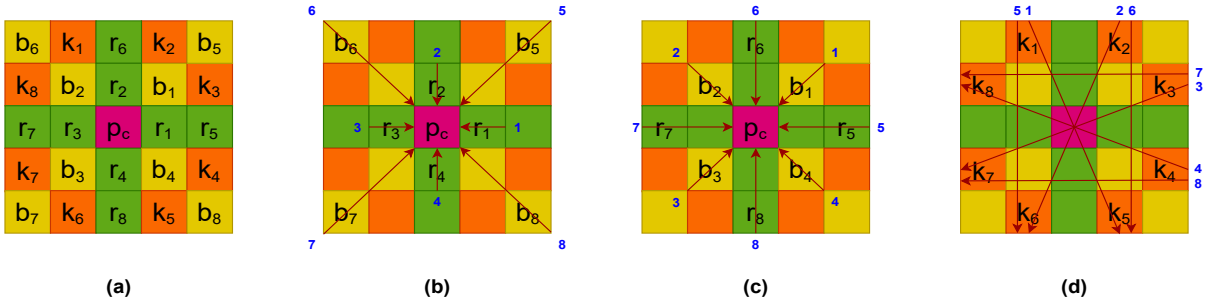


Figure 4.7: Process of feature extraction through RCP, CSP and RCSP (a) Numbering scheme of chessmen followed for feature extraction (b) Feature extraction through RCP_1 (c) Feature extraction through RCP_2 and (d) Feature extraction through CSP.

corresponding equations for calculating RCP_1 are shown in eqs.(4.18) and (4.19).

Generally, weight matrix contains binary weights [17, 46, 97]. Upon using binary weights (shown in eq.(3.27)), the fv length of RCP_1 is 256, as like LBP. To further reduce the fv length, different weights such as fibonacci (shown in eq.(3.28)) [164], prime (shown in eq.(3.29)), natural (shown in eq.(3.30)), squares (shown in eq.(3.31)) and odd (shown in eq.(3.32)) have been considered. For prime weights, the sequence of first eight prime numbers are considered, which is shown in eq.(3.29)). Similarly, for other weights also, the sequence of first eight numbers in that particular series are considered. At each time, w_m can take any one the weight values for feature extraction. The number thus obtained after multiplying with w_m is then replaced in the value of pixel p_c.

$$RCP_1 = \{s(r_1, p_c), s(r_2, p_c), s(r_3, p_c), s(r_4, p_c), s(b_5, p_c), s(b_6, p_c), s(b_7, p_c), s(b_8, p_c)\} \tag{4.18}$$

$$RCP_1 = \sum (RCP_1 . * w_m) \tag{4.19}$$

4.3.2 RCP_2

RCP_2 contains a set of eight pixels, which includes four pixels corresponding to Bishop, considered from the 3 x 3 neighborhood (rd = 1) and the remaining four pixels corresponding to Rook, considered from the 5 x 5 neighborhood (rd = 2). The pixel positions corresponding to RCP_2 are shown in figure 4.7 (c). Thus, considering pixels in this manner enables in better capturing the information in eight directions ($45°$, $135°$, $225°$, $315°$, $0°$,

90°, 180°, 270°) respectively. The pixel intensities present in these eight positions ($b_{1,2,3,4}$, $r_{5,6,7,8}$) are compared with the intensity of pixel p_c. Upon comparison, if the obtained result is positive, then the corresponding bit is encoded as one, else it encoded as zero. Thus, for eight pixel positions, eight corresponding binary values are obtained, which are then concatenated to form an eight bit binary number is subsequently multiplied with the w_m. The number thus obtained after multiplying with w_m is then replaced with the value of pixel p_c. The corresponding equations for calculating fv based on RCP_2 are shown in eqs.(4.20) and (4.21).

$$RCP_2 = \{s(b_1, p_c), s(b_2, p_c), s(b_3, p_c), s(b_4, p_c), s(r_5, p_c), s(r_6, p_c), s(r_7, p_c), s(r_8, p_c)\} \tag{4.20}$$

$$RCP_2 = \sum(RCP_2. * w_m) \tag{4.21}$$

Algorithm 4.2 Feature Extraction through RCP

Input: An Input image (Img) of size N x N
Output: Feature vector (fv_{rcp}) of size ⌈(N-4)/C⌉ * ⌈(N-4)/C⌉ * 2 * L

procedure RCP(Img)
  Initialization: RCP_1, RCP_2, RCP ← { }
  Load an input image (Img)
  for all a ∈ range(1,N-4) **do**
    for all b ∈ range(1,N-4) **do**
      Block = img(a:a+4,b:b+4)
      Assign the pixel values to Rook, Bishop and Knight in the 5 x 5 block.
      Calculate RCP_1 using eqs.(4.18) and (4.19)
      Calculate RCP_2 using eqs.(4.20) and (4.21)
    end for
  end for
  Create two feature response maps obtained from RCP_1 and RCP_2 by reshaping them to (N-4) x (N-4).
  Each feature response map is partitioned into C x C non-overlapping blocks.
  Histograms are extracted block wise for both the feature response maps using eqs.(4.22) and (4.23).
  Concatenate all the histograms to obtain feature vector fv_{rcp} using eq.(4.24).
  return fv_{rcp}
end procedure

4.3.3 Feature Level Fusion

RCP is obtained by horizontally concatenating both RCP_1 and RCP_2. Thus, RCP feature descriptor generates two feature codes, each corresponding to RCP_1 and RCP_2 and hence, the fv length of RCP becomes 512 (incase of binary weights). The fv length becomes 110, 156, 74, 410, 130 and 146 whenever fibonacci, prime, natural, squares, odd and even weights are utilized for feature extraction. Thus, by using other weights than binary weights, even if two feature codes are generated, the fv length of RCP is much lesser than fv generated for one feature code (LBP, LDP) in all cases (expect whenever squares weights are used for feature extraction). In eqs.(4.22) and (4.23), $hist_{rcp_1}$ and $hist_{rcp_2}$ corresponds to the block wise histograms of RCP_1 and RCP_2 respectively. The corresponding equation for calculating features (fv_{rcp}) extracted using RCP is shown in eq.(4.24). The algorithm for feature extraction through RCP is mentioned in algorithm 4.2. The value of 'L' in algorithm 4.2 depends on the weight matrix used in the process of feature extraction. For an input image of size N x N, the computational complexity of the proposed RCP method is $O(N^2)$. The feature response maps generated by RCP_1 and RCP_2 methods using binary weights is shown in figure 4.8(a-c).

$$hist_{rcp_1} = Hist(RCP_1) \tag{4.22}$$

$$hist_{rcp_2} = Hist(RCP_2) \tag{4.23}$$

$$fv_{rcp} = hist_{rcp_1} \cup hist_{rcp_2} \tag{4.24}$$

4.4 Chess Symmetric Pattern (CSP) Feature Extraction

The process of feature extraction through CSP is inspired from LGC-AD operator. The length of fv generated by LGC-AD is 4096 (very high) and also the computational complexity involved in LGC-AD is very high. From LGC-AD, the concept of comparing the horizontal, veritical and diagonal pixel information is adopted while designing the CSP operator. The numbering assignment of Knight is done in clockwise manner. CSP operator captures the pixel information in four diagonal directions, two vertical directions and in two horizontal directions. Thus, by comparing the pixels as shown in figure 4.7(d), the fv length

Algorithm 4.3 Feature Extraction through CSP

Input: An Input image (Img) of size N x N
Output: Feature vector (fv_{csp}) of size ⌈(N-4)/C⌉ * ⌈(N-4)/C⌉ * L

procedure CSP(Img)
  Initialization: CSP ← { }
  Load an input image (Img)
  for all a ∈ range(1,N-4) **do**
    for all b ∈ range(1,N-4) **do**
      Block = img(a:a+4,b:b+4)
      Assign the pixel values to Rook, Bishop and Knight in the 5 x 5 block.
      Calculate CSP using eqs.(4.25) and (4.26)
    end for
  end for
  Create a feature response map obtained from CSP by reshaping it into (N-4) x (N-4).
  The feature response map is partitioned into C x C non-overlapping blocks.
  Histograms are extracted block wise from the feature response map to obtain feature vector fv_{csp} as shown in eq.(4.27).
  return fv_{csp}
end procedure

of CSP is 16 times lesser than the fv length of LGC-AD. The corresponding equations for calculating CSP is shown in eq.(4.25) and eq.(4.26). In eq.(4.27), fv_{csp} corresponds to the block wise histograms extracted using CSP. The algorithm for feature extraction through CSP is mentioned in algorithm 4.3. The value of 'L' in algorithm 4.3 depends on the weight matrix used in the process of feature extraction. For an input image of size N x N, the computational complexity of the proposed CSP method is $O(N^2)$. The feature response maps generated by CSP method using binary weights is shown in figure 4.8(d).

$$CSP = \{s(k_1, k_5), s(k_2, k_6), s(k_3, k_7), s(k_4, k_8), s(k_1, k_6), s(k_2, k_5), s(k_3, k_8), s(k_4, k_7)\} \tag{4.25}$$

$$CSP = \sum(CSP. * w_{\mathrm{m}}) \tag{4.26}$$

$$fv_{csp} = Hist(CSP) \tag{4.27}$$

Figure 4.8: (a) Happy expression image from TFEID dataset. Feature response maps generated by (b) RCP_1 (c) RCP_2 and (c) CSP using binary weights.

4.5 Radial Cross Symmetric Pattern (RCSP) Feature Extraction

RCSP is obtained by horizontally concatenating both RCP and CSP. Thus, RCSP generates three feature codes, two feature codes corresponding to RCP_1 and RCP_2 and one feature code corresponding to CSP. Hence, the fv length of RCSP becomes 768 (incase of binary weights). But, as other weights are used, the fv length becomes 165, 234, 111, 615 and 195 whenever fibonacci, prime, natural, squares, and odd weights are utilized for feature extraction. Thus, by using other weights than binary weights, even if three feature codes are generated, the fv length of RCSP is lesser than fv generated for one feature code (LBP, LDP) in all cases (expect whenever squares weights are used for feature extraction). The corresponding equation for computing features using RCSP is shown in eq.(4.28).

$$fv_{rcsp} = hist_{rcp} \cup hist_{csp} \tag{4.28}$$

4.6 Results and Comparison Analysis

In this section, the feature vector length comparison of proposed feature descriptors with different weights, the experimental results and the comparison of proposed methods with the existing methods is reported. For the purpose of classification, a multi-class SVM classifier with a linear kernel is employed for classifying the query images into various expressions.

Table 4.1: Feature vector length comparisons of proposed methods

Method	Binary	Fibonacci	Prime	Natural	Squares	Odd
RMP	512	110	156	74	410	130
RCP	512	110	156	74	410	130
CSP	256	55	78	37	205	65
RCSP	768	165	234	111	615	195

4.6.1 Feature Vectors comparison

For each of the proposed methods, the feature vector length comparison using different weights is shown in table 4.1. RMP and RCP methods generate two feature codes each, hence, their fv length is twice the length of LBP (in case of binary weights). As CSP method generates only feature code, it's fv length is same as that of LBP. The fv length of RCSP is thrice the length of LBP (in case of binary weights), as RCSP extracts three features in a local neighborhood. Thus, to the proposed feature descriptors, different weights have been applied to effectively reduce the fv length and to find out the optimal recognition accuracy.

4.6.2 Experiments for Six Expressions

The experiments for six expressions have been conducted on different 'in the lab' datasets. The recognition accuracy comparison analysis using RP, MP and RMP is shown in table 4.2 for six expressions. The experimental results from table 4.2 demonstrated that RP contributes more when compared to MP in RMP. Although, RP contributed more, the information captured by MP cannot be completely ignored, as it captures discriminative information with respect to Knight pixel positions. Also, the experimental results demonstrated that considering fusion of RP and MP, resulted in an enhanced accuracy, rather than considering RP or MP alone. So, in this Chapter, RMP, fusion of RP and MP is employed to better capture minute changes with respect to facial expressions in a local neighborhood. The recognition accuracy of RMP for different weights and block sizes (C x C) on MUG dataset for six expression classification is shown in figure 4.9. From figure 4.9, it is observed that the highest recognition accuracy of 88.22% is obtained by using prime weights

Table 4.2: Recognition accuracy comparison analysis using RP, MP and RMP for six expressions

Dataset	**JAFFE**			**TFEID**			**MUG**			**OULU**		
Method	**RP**	**MP**	**RMP**	**RP**	**MP**	**RMP**	**RP**	**MP**	**RMP**	**RP**	**MP**	**RMP**
Binary	57.31	57.69	58.27	93.92	94.17	94.67	86.74	86.89	86.96	75.76	75.42	75.90
Fibonacci	58.42	58.42	61.17	94.67	94.58	95.00	87.19	85.63	87.19	75.83	75.42	75.42
Prime	60.01	60.40	61.64	94.08	93.67	94.17	86.89	86.30	88.22	75.56	75.90	76.18
Natural	56.73	56.75	60.00	93.25	93.33	93.67	84.59	83.70	86.52	73.89	73.75	75.49
Squares	56.73	55.64	58.77	93.42	94.08	94.25	86.96	87.48	88.15	76.04	75.97	76.18
Odd	58.78	59.91	60.03	94.25	94.08	94.67	87.04	85.33	87.11	75.35	74.70	75.56

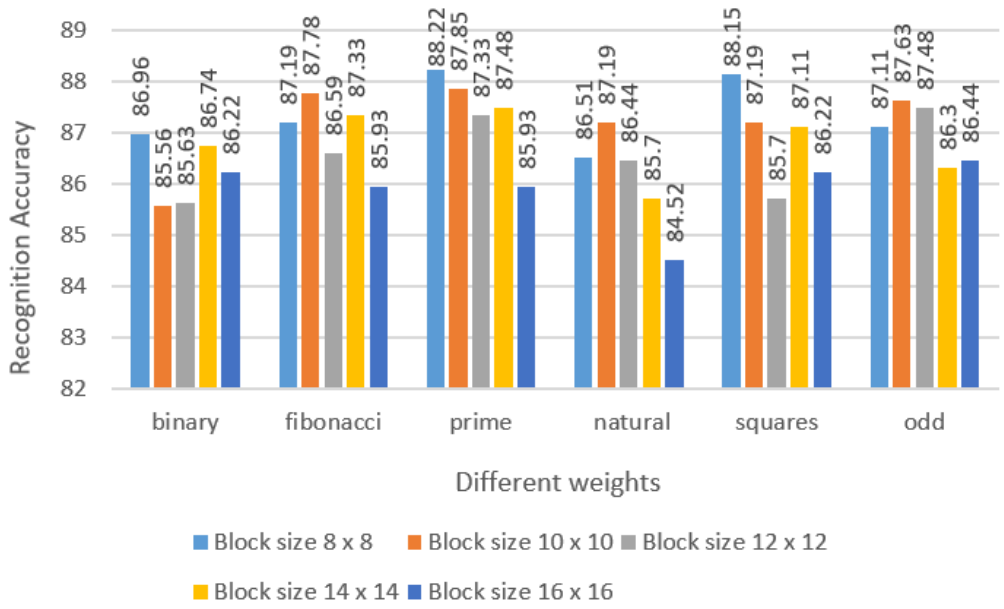


Figure 4.9: Recognition accuracy of RMP for different weights and block sizes (C x C) on MUG dataset for six expressions classification

for block size of 8 x 8. Thus, in order to maintain an uniformity across datasets, this block size of 8 x 8 is chosen for experimental analysis.

The proposed methods have been implemented with different weights and the results have been tabulated. In table 4.3, for each dataset, the recognition accuracy comparison of RMP, in table 4.4, the recognition accuracy comparison of RCP, in table 4.5, the recognition accuracy comparison of CSP, and in table 4.6, the recognition accuracy comparison of RCSP with different weights is shown. Among the proposed methods with different weights, CSP method with fibonacci weights achieved an optimal recognition accuracy of

Table 4.3: Recognition accuracy of RMP with different weights for six expressions on different 'in the lab' datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	58.27	61.17	61.64	60	58.77	60.03
MUG	86.96	87.19	88.22	86.52	88.15	87.11
CK+	91.45	91.67	91.23	91.33	91.34	91.33
OULU	75.90	75.42	76.18	75.49	76.18	75.56
TFEID	94.67	95	94.17	93.67	94.25	94.67
KDEF	82.86	82.86	83.80	83.57	83.10	82.86
WSEFEP	87.78	87.78	87.22	87.22	87.78	88.33
ADFES	90.91	93.18	90.91	90.15	90.91	90.91

Table 4.4: Recognition accuracy of RCP with different weights for six expressions on different 'in the lab' datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	59.48	60.90	58.92	59.94	57.60	61.51
MUG	87.70	87.41	87.93	88.07	87.85	86.52
CK+	91.99	91.88	91.66	91.44	92.21	90.67
OULU	75.97	75.14	76.18	74.51	75.56	75.49
TFEID	95.08	94.58	95.50	95.42	95.58	94.17
KDEF	84.05	83.10	83.33	84.52	83.33	83.57
WSEFEP	87.22	87.22	88.33	89.44	87.22	86.67
ADFES	92.27	92.42	91.67	91.67	93.94	93.18

Table 4.5: Recognition accuracy of CSP with different weights for six expressions on different 'in the lab' datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	62.70	63.66	61.61	62.08	61.52	62.64
MUG	86.22	86.59	85.59	86.59	86.44	86.44
CK+	90.79	90.70	91.13	91.02	90.79	90.69
OULU	73.33	73.54	73.13	72.92	73.40	73.68
TFEID	93.50	95.08	93.75	95.08	92.83	94.25
KDEF	82.86	83.57	82.14	82.38	82.62	83.33
WSEFEP	85.56	87.22	87.22	86.11	86.67	86.11
ADFES	92.42	90.91	90.91	90.91	92.42	92.42

Table 4.6: Recognition accuracy of RCSP with different weights for six expressions on different 'in the lab' datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	63.08	60.00	59.97	60.06	59.97	61.11
MUG	86.59	87.63	88.00	88.07	87.59	87.26
CK+	92.21	91.99	91.88	91.99	92.42	91.77
OULU	74.23	74.30	74.31	75.10	74.30	74.86
TFEID	95.17	94.58	94.58	95	95.08	95
KDEF	83.81	83.57	83.33	84.05	83.57	83.33
WSEFEP	87.22	87.22	88.33	88.89	86.67	87.78
ADFES	91.67	92.42	92.42	91.67	93.18	91.67

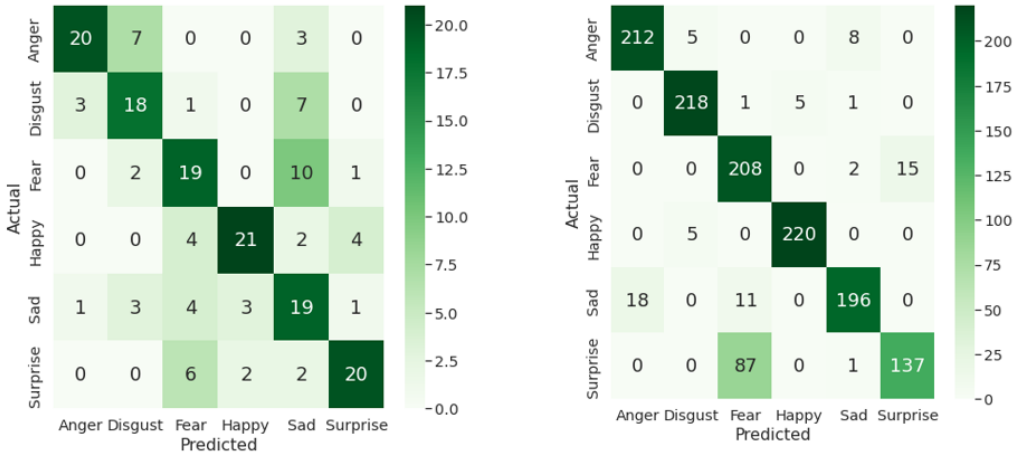


Figure 4.10: Confusion matrix for six expressions on (a) JAFFE dataset using CSP method with fibonacci weights (b) MUG dataset using RMP method with prime weights

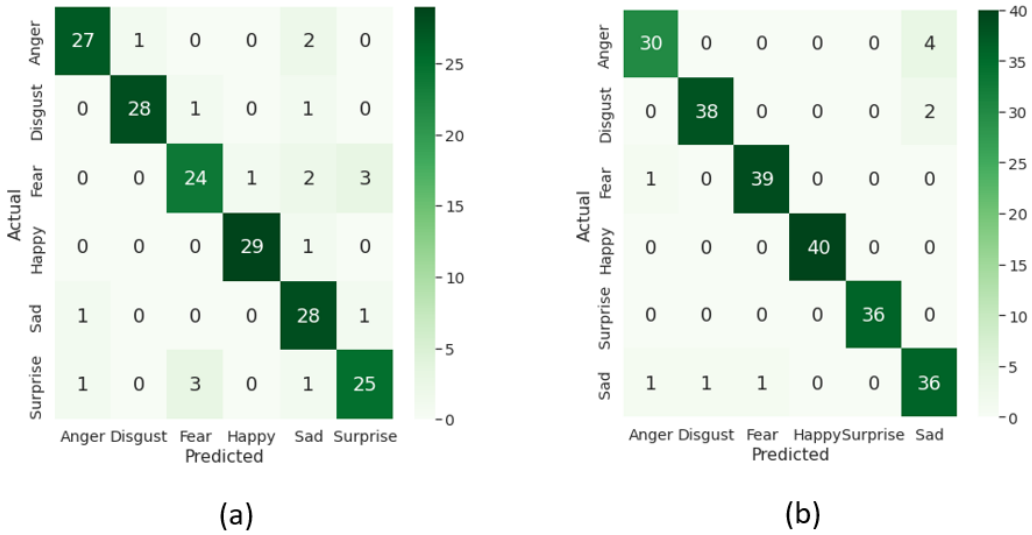


Figure 4.11: Confusion matrix for six expressions on (a) WSEFEP dataset using RCP method with natural weights (b) TFEID dataset using RCP method with squares weights

63.66% on JAFFE dataset. RMP method with prime weights achieved an optimal recognition accuracy of 88.22% and 76.18% on MUG and OULU datasets respectively. The proposed RCSP method with squares weights achieved an optimal recognition accuracy of 92.42% on CK+ dataset.

In case of TFEID and ADFES datasets, RCP method with squares weights achieved an optimal recognition accuracy of 95.58% and 93.94% respectively. In case of KDEF and WSEFEP datasets, RCP method with natural weights achieved an optimal recognition accuracy of 84.52% and 89.44% respectively. The confusion matrix obtained using CSP method with fibonacci weights for JAFFE dataset is presented in figure 4.10(a) and for MUG dataset using RMP method with prime weights is presented in figure 4.10(b). The confusion matrix obtained using RCP method with natural weights for WSEFEP dataset is presented in figure 4.11(a) and for TFEID dataset using RCP method with squares weights is presented in figure 4.11(b). In table 4.7, the comparison analysis of the proposed methods with the existing variants of binary patterns, implemented in our environment setup is shown. In table 4.8, the comparison analysis of the proposed methods with the existing methods is shown. In tables 4.7 and 4.8, the proposed methods and their recognition accuracy has been highlighted in bold. In figure 4.12, the comparison analysis of proposed method with existing variants of binary patterns on MUG and KDEF datasets is shown. From table 4.7, the proposed methods outperformed the existing variants of binary patterns on all other datasets except KDEF dataset. Although LDDSCP method achieved 0.24% more than the proposed method in case of KDEF dataset, the proposed methods are better as they achieved better results on other datasets also. From table 4.8, for different 'in the lab' datasets, the proposed methods outperformed the existing FER methods.

4.6.3 Experiments for Seven Expressions

The experiments for seven expressions have been conducted on all the ten datasets for RMP, RCP, CSP and RCSP methods. The proposed methods have been implemented with different weights and the results have been tabulated. In table 4.9, for each dataset, the recognition accuracy comparison of RMP, in table 4.10, the recognition accuracy comparison of RCP, in table 4.11, the recognition accuracy comparison of CSP and in table 4.12,

Table 4.7: Comparison analysis with existing variants of binary patterns for six expressions on different 'in the lab' datasets

Method	JAFFE	MUG	CK+	OULU	TFEID	KDEF	WSEFEP	ADFES
LBP [46]	56.66	82.65	89.97	75.34	92.42	80.95	87.22	90.16
LDP [97]	52.77	82.87	90.84	72.43	93.67	80.95	86.67	90.91
LDN [31]	56.66	81.96	88.84	72.29	93.33	82.62	87.22	88.64
CSLBP [28]	53.28	83.94	90.68	71.53	94.25	81.67	84.44	87.12
LGC [84]	58.89	86.22	89.61	73.47	95.00	83.81	87.78	90.91
LDTP [32]	55.55	82.04	89.60	72.29	93.25	83.57	87.78	88.64
LDTeRP [33]	58.54	80.15	85.89	68.54	90.75	81.43	81.11	84.89
ALDP [35]	55.09	82.89	89.06	70.28	93.54	80.95	85.00	85.61
MSBP [88]	56.75	85.78	90.58	73.41	94.25	83.10	86.67	90.15
LDSP [34]	56.70	85.19	91.54	68.47	94.50	82.38	85.00	86.36
LDDSCP [99]	59.55	86.15	89.61	73.47	95.50	84.76	83.89	89.39
RADAP [17]	57.22	83.48	90.63	75.90	94.17	82.38	87.78	91.67
LBP + LNeP [134]	61.29	86.52	92.21	75.00	93	83.57	88.89	90.15
RMP	**61.64**	**88.22**	**91.67**	**76.18**	**95**	**83.80**	**88.33**	**93.18**
RCP	**61.51**	**88.07**	**92.21**	**76.18**	**95.58**	**84.52**	**89.44**	**93.94**
CSP	**63.66**	**86.89**	**91.13**	**73.82**	**95.08**	**83.57**	**87.22**	**92.42**
RCSP	**63.08**	**88.07**	**92.42**	**75.10**	**95.17**	**84.05**	**88.89**	**93.18**

Table 4.8: Comparison with existing methods for six expressions on different 'in the lab' datasets

Dataset	Method	Accuracy	Dataset	Method	Accuracy
JAFFE	IFRBC [86]	62.29	MUG	ResNet50 [17]	86.88
	ResNet50 [17]	59.44		DAGSVM [129]	82.28
	LOOP [11]	58.72		LOOP [11]	85.33
	WLGC-HD [57]	60.7		HiNet [108]	87.8
	RMP	**61.64**		**RMP**	**88.22**
	RCP	**61.51**		**RCP**	**88.07**
	CSP	**63.66**		**CSP**	**86.89**
	RCSP	**63.08**		**RCSP**	**88.07**
CK+	ResNet50 [17]	89.32	OULU	ResNet50 [17]	73.1
	HiNet [108]	91.40		HiNet [108]	74.3
	WLGC-HD [57]	72.80		VGG16 [17]	73.4
	RMP	**91.67**		**RMP**	**76.18**
	RCP	**92.21**		**RCP**	**76.18**
	CSP	**91.13**		**CSP**	**73.82**
	RCSP	**92.42**		**RCSP**	**75.10**
TFEID	DSNGE [85]	93.89	KDEF	IFRBC [86]	77.98
	DAMCNN [102]	93.65		ICVR [86]	76.31
	LIoP + HOG [123]	93.50		HOG [108]	82.17
	RMP	**95**		**RMP**	**83.80**
	RCP	**95.58**		**RCP**	**84.52**
	CSP	**95.08**		**CSP**	**83.57**
	RCSP	**95.17**		**RCSP**	**84.05**
WSEFEP	LOOP [11]	87.78	ADFES	LOOP [11]	91.67
	RMP	**88.33**		**RMP**	**93.18**
	RCP	**89.44**		**RCP**	**93.94**
	CSP	**87.22**		**CSP**	**92.42**
	RCSP	**88.89**		**RCSP**	**93.18**

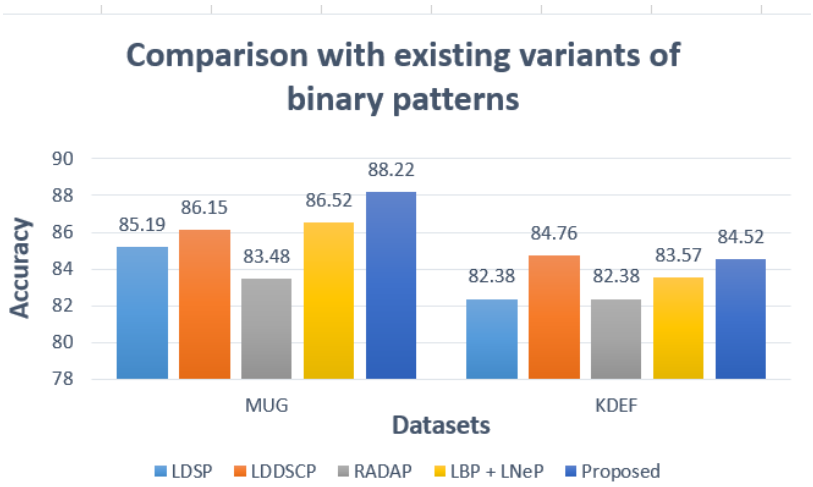


Figure 4.12: Comparison analysis of proposed method with existing variants of binary patterns for six expressions on MUG and KDEF datasets

the recognition accuracy comparison of RCSP with different weights is shown. Among the proposed methods with different weights, CSP method with fibonacci weights achieved an optimal recognition accuracy of 62.19% on JAFFE dataset. RCP method with natural weights achieved an optimal recognition accuracy of 96.19% and 85.71% on TFEID and WSEFEP datasets respectively. The proposed RCP method with binary and odd weights achieved an optimal recognition accuracy of 75.71% and 94.16% on OULU and ADFES datasets respectively. In case of MUG, CK+ and KDEF datasets, RCSP method with natural, squares and prime weights achieved an optimal recognition accuracy of 83.75%, 88% and 82.65% respectively. RCSP method with fibonacci and natural weights achieved an optimal recognition accuracy of 77.64% and 99.99% on RAF and FERG datasets respectively.

The confusion matrix obtained using RCSP method with squares weights for CK+ dataset is presented in figure 4.13(a) and for KDEF dataset using RCSP method with prime weights is presented in figure 4.13(b). In table 4.13, the comparison analysis of the proposed methods with the existing variants of binary patterns, implemented in our environment setup is shown. In table 4.14, the comparison analysis of the proposed methods with the existing methods is shown. The comparison analysis for RAF and FERG datasets with the existing methods is reported in table 4.15. In tables 4.13, 4.14 and 4.15, the proposed methods and their recognition accuracy has been highlighted in bold. From

Table 4.9: Recognition accuracy of RMP with different weights for seven expressions on different datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	59.03	60.02	59.45	57.07	59.43	58.20
MUG	82.60	82.35	82.92	82.54	83.49	82.22
CK+	86.55	86.46	86.38	86.41	86.55	86.20
OULU	74.87	74.47	75.24	74.76	74.76	75.24
TFEID	93.99	94.64	93.63	93.57	93.99	93.63
KDEF	81.22	81.43	81.63	81.43	81.63	81.63
WSEFEP	84.76	83.81	84.76	84.29	85.24	85.24
ADFES	93.51	93.51	92.21	93.51	92.21	92.86
RAF	75.68	75.85	76.96	74.61	75.65	75.98
FERG	99.11	98.17	99.23	99.43	98.23	98.26

Table 4.10: Recognition accuracy of RCP with different weights for seven expressions on different datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	58.09	60.36	58.04	56.22	58.00	57.56
MUG	80.25	83.62	83.68	82.48	82.54	82.03
CK+	85.98	85.96	85.88	86.02	85.98	86.11
OULU	75.71	75.24	74.87	74.76	74.67	74.76
TFEID	94.36	93.99	95.12	96.19	94.76	93.99
KDEF	81.43	81.22	81.43	82.45	81.02	81.02
WSEFEP	84.29	84.29	84.76	85.71	84.29	83.33
ADFES	93.40	91.56	92.21	92.21	94.16	94.16
RAF	76.04	76.80	76.66	75.75	77.02	75.13
FERG	99.83	98.73	98.74	99.90	99.21	98.47

table 4.13, the proposed methods outperformed the existing variants of binary patterns on all other datasets except WSEFEP dataset. From table 4.14, the proposed methods outperformed the existing FER methods on all other datasets except MUG and CK+ datasets. In case of MUG dataset, HiNet and ResNet50 methods achieved 3.45% and 1.83% more than the proposed RCSP method. In case of CK+ dataset, HiNet method achieved 0.6% better recognition accuracy than the proposed RCSP method. The proposed methods are simple when compared to HiNet and ResNet50 methods which contains one million and thirty one million parameters respectively. From table 4.15, the proposed RCSP method with fibonacci weights and natural weights outperformed the existing methods on RAF and FERG datasets respectively.

Table 4.11: Recognition accuracy of CSP with different weights for seven expressions on different datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	61.32	62.19	61.80	61.73	60.32	60.35
MUG	80.57	81.65	81.59	80.95	81.08	81.90
CK+	86.86	86.55	86.78	86.68	86.61	85.85
OULU	72.95	73.35	72.86	72.86	73.81	72.95
TFEID	94.64	94.29	95.00	94.29	94.29	94.64
KDEF	81.02	80.82	80.61	80.82	81.22	80.82
WSEFEP	84.29	84.29	83.81	83.81	84.29	84.29
ADFES	91.56	92.21	90.91	90.91	91.56	91.56
RAF	72.36	71.31	72.62	71.41	72.62	72.88
FERG	99.99	99.71	99.97	99.51	99.87	99.10

Table 4.12: Recognition accuracy of RCSP with different weights for seven expressions on different datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	60.97	60.91	61.00	61.04	60.00	60.50
MUG	82.54	83.57	83.30	83.75	82.98	83.49
CK+	87.68	87.30	87.62	87.61	88.00	87.24
OULU	74.76	73.96	74.45	73.45	73.66	73.96
TFEID	94.40	93.99	94.36	95.06	94.90	94.70
KDEF	82.65	82.24	82.65	82.45	82.65	81.63
WSEFEP	83.81	84.76	84.29	85.71	83.33	85.24
ADFES	92.86	92.21	92.21	91.56	93.51	91.56
RAF	76.27	77.64	77.15	76.56	77.12	76.89
FERG	99.99	99.46	99.44	99.99	99.97	99.87

Table 4.13: Comparison analysis with existing variants of binary patterns for seven expressions on different 'in the lab' datasets

Method	JAFFE	MUG	CK+	OULU	TFEID	KDEF	WSEFEP	ADFES
LBP [46]	53.65	76.16	83.96	65.71	92.02	78.16	82.38	87.66
LDP [97]	52.10	78.70	84.80	69.16	86.20	80.61	80.95	90.26
LDN [31]	54.87	77.85	83.35	70.89	93.51	80.75	83.81	91.56
CSLBP [28]	52.40	79.37	85.51	57.98	89.44	80.20	79.44	86.36
LGC [84]	57.59	82.03	86.71	71.13	93.43	81.02	84.29	90.91
LDTP [32]	51.32	78.70	83.08	68.86	93.15	80.81	85.71	85.71
LDTeRP [33]	51.70	78.11	81.40	64.53	90.18	77.35	79.52	79.87
ALDP [35]	51.53	77.59	85.61	67.74	92.36	76.53	80.48	83.77
MSBP [88]	56.81	81.40	86.04	73.75	93.27	81.22	83.73	91.56
LDSP [34]	52.49	80.63	84.19	63.81	93.15	80.61	85.00	85.06
LDDSCP [99]	57.37	80.95	84.93	69.57	90.95	81.84	82.38	88.96
RADAP [17]	56.20	80.26	84.60	74.34	93.27	80.20	87.42	90.91
LBP + LNeP [134]	59.04	81.45	86.30	73.20	93.27	81.84	86.19	90.26
RMP	**60.02**	**83.49**	**86.55**	**75.24**	**94.64**	**81.63**	**85.24**	**93.51**
RCP	**60.36**	**83.68**	**86.11**	**75.71**	**96.19**	**82.45**	**85.71**	**94.16**
CSP	**62.19**	**81.90**	**86.86**	**73.81**	**95**	**81.22**	**84.29**	**92.21**
RCSP	**61.04**	**83.75**	**88**	**74.45**	**95.10**	**82.65**	**85.71**	**93.51**

Table 4.14: Comparison with existing methods for seven expressions on different 'in the lab' datasets

Dataset	Method	Accuracy	Dataset	Method	Accuracy
JAFFE	PCANet [117]	58.35	MUG	CBA [90]	78.57
	ResNet50 [17]	57.13		ResNet50 [17]	85.58
	LOOP [11]	59.67		LOOP [11]	79.68
	WLGC-HD [57]	58.2		HiNet [108]	87.2
	RMP	**60.02**		**RMP**	**83.49**
	RCP	**60.36**		**RCP**	**83.68**
	CSP	**62.19**		**CSP**	**81.90**
	RCSP	**61.04**		**RCSP**	**83.75**
CK+	ResNet50 [17]	87.31	OULU	ResNet50 [17]	65.4
	HiNet [108]	88.6		HiNet [108]	72
	DLFS [107]	83.72		VGG19 [17]	70.5
	RMP	**86.55**		**RMP**	**75.24**
	RCP	**86.11**		**RCP**	**75.71**
	CSP	**86.86**		**CSP**	**73.81**
	RCSP	**88**		**RCSP**	**74.45**
TFEID	DAMCNN [102]	93.36	KDEF	DLFS [107]	78.60
	Pyramid+SBDT [87]	93.38		PCANet [117]	69.59
	MSDV [132]	93.50		SAFL [108]	81.22
	RMP	**94.64**		**RMP**	**81.63**
	RCP	**96.19**		**RCP**	**82.45**
	CSP	**95**		**CSP**	**81.22**
	RCSP	**95.10**		**RCSP**	**82.65**
WSEFEP	LOOP[11]	84.68	ADFES	AFM [103]	92.70
	RMP	**85.24**		**RMP**	**93.51**
	RCP	**85.71**		**RCP**	**94.16**
	CSP	**84.29**		**CSP**	**92.21**
	RCSP	**85.71**		**RCSP**	**93.51**

Table 4.15: Comparison with existing methods for seven expressions on RAF and FERG datasets

Dataset	**Method**	**Accuracy**	**Dataset**	**Method**	**Accuracy**
RAF	DLPCNN [55]	74.20	FERG	Deep Expr [56]	89.02
	ICID Fusion [101]	75.40		Ensemble Multi-Feature [17]	97
	Sadeghi et al. [93]	76.23		Adversarial NN [105]	98.2
	DCNN+RLPS [108]	72.84		Deep Emotion [106]	99.3
	IFSL [120]	76.9		LBP-AW [16]	96.7
	RMP	**76.96**		**RMP**	**99.43**
	RCP	**77.02**		**RCP**	**99.90**
	CSP	**72.88**		**CSP**	**99.99**
	RCSP	**77.64**		**RCSP**	**99.99**

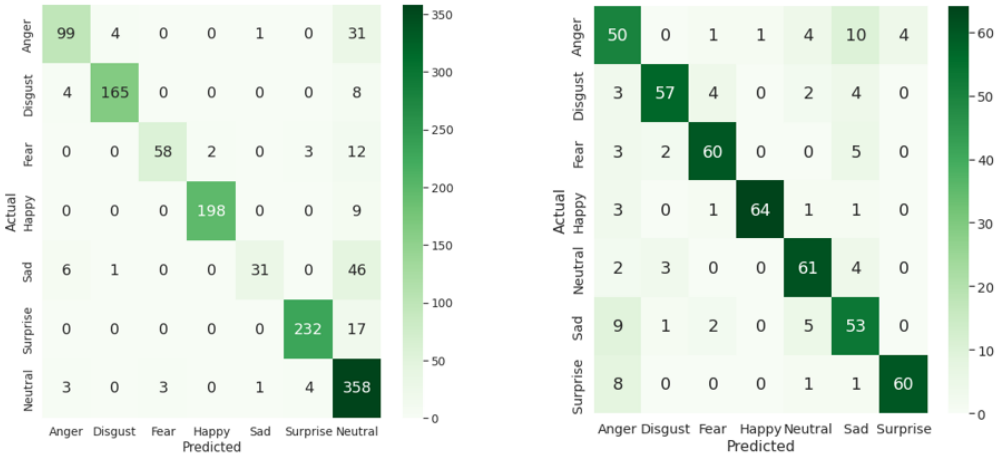


Figure 4.13: Confusion matrix for seven expressions on (a) CK+ dataset using RCSP method with squares weights (b) KDEF dataset using RCSP method with prime weights

Table 4.16: Recognition accuracy comparison for eight expressions with different weights on TFEID Dataset

Method	Binary	Fiboancci	Prime	Natural	Squares	Odd
RMP	91.31	91.31	91.34	90.80	91.31	90.80
RCP	91.14	91.14	91.14	91.32	91.17	91.80
CSP	93.85	93.85	94.13	93.54	93.85	93.85
RCSP	92.91	92.32	92.28	92.32	92.32	92.32

4.6.4 Experiments for Eight Expressions

The proposed methods have been implemented with different weights and the results have been tabulated in table 4.16. Among the proposed methods with different weights, CSP method with prime weights achieved an optimal recognition accuracy of 94.13%. The confusion matrix obtained using CSP method with prime weights on TFEID dataset for eight expressions is presented in figure 4.14. The comparison analysis of the proposed methods with the existing variants of binary patterns is reported in the second column of table 4.18. In table 4.18, the proposed methods and their recognition accuracy has been highlighted in bold. The experimental results from table 4.18 indicate that the proposed methods outperformed the existing variants of binary patterns in terms of recognition accuracy.

Table 4.17: Recognition accuracy comparison for ten expressions with different weights on ADFES Dataset

Method	Binary	Fiboancci	Prime	Natural	Squares	Odd
RMP	88.43	87.19	87.02	87.42	87.98	85.71
RCP	85.20	86.22	86.97	86.02	88.03	86.11
CSP	85.01	84.96	84.05	83.54	85.01	85.01
RCSP	85.41	86.57	87.02	85.41	87.12	86.62

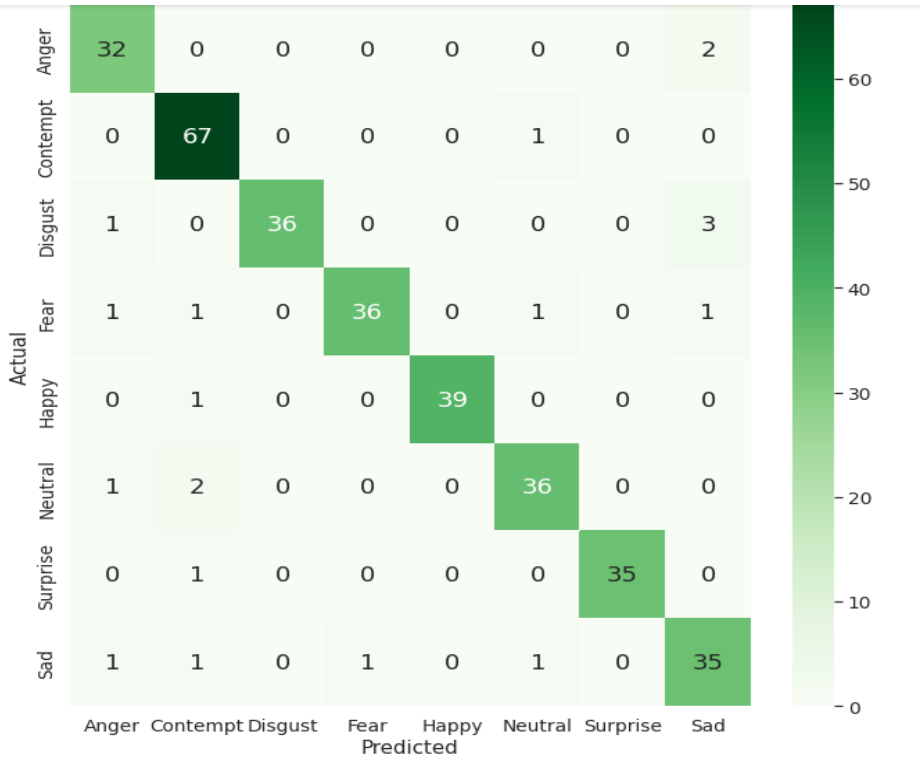


Figure 4.14: Confusion matrix for eight expressions on TFEID dataset using CSP method with prime weights

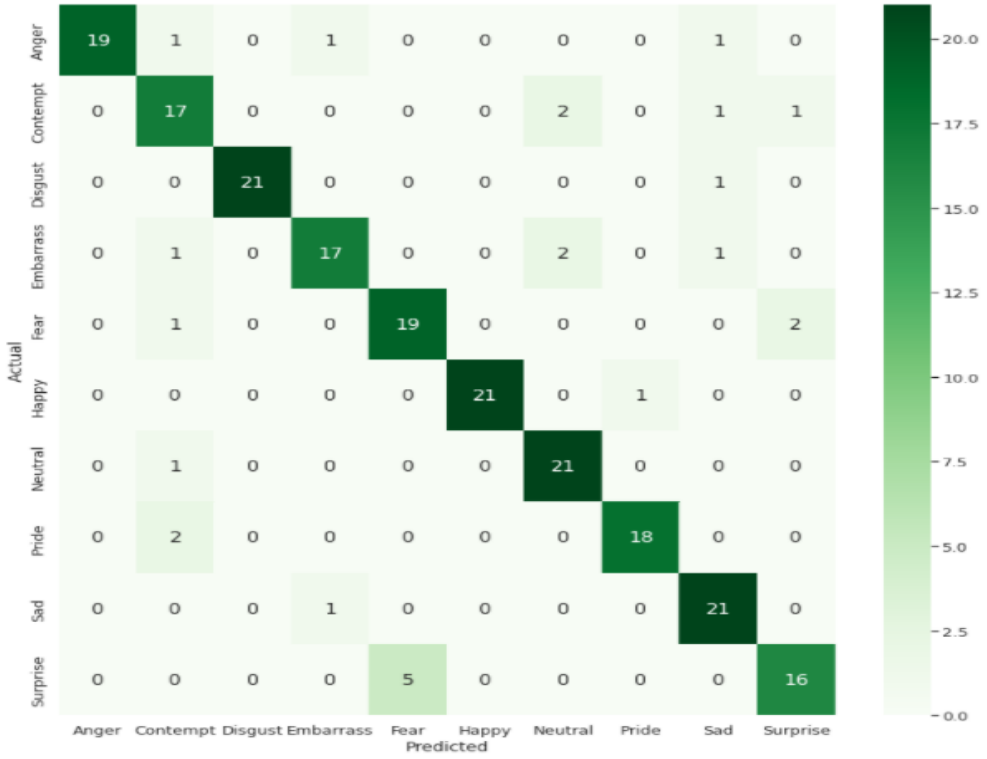


Figure 4.15: Confusion matrix for ten expressions on ADFES dataset using RMP method with binary weights

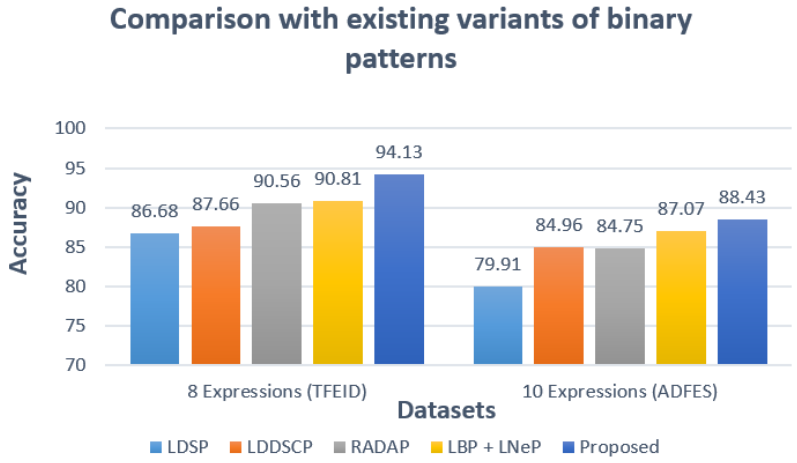


Figure 4.16: Comparison analysis of proposed method with existing variants of binary patterns for eight and ten expressions on TFEID and ADFES datasets

Table 4.18: Comparison analysis with existing variants of binary patterns for eight and ten expressions on TFEID and ADFES datasets

Method	Eight Expressions	Ten Expressions
LBP [46]	90.95	83.54
LDP [97]	91.17	85.25
LDN [31]	88.06	85.20
CSLBP [28]	89.45	80.61
LGC [84]	89.95	85.20
LDTP [32]	86.29	82.68
LDTerP [33]	88.59	76.53
ALDP [35]	90.36	78.40
MSBP [88]	89.54	87.17
LDSP [34]	86.68	79.91
LDDSCP [99]	87.66	84.96
RADAP [17]	90.56	84.75
LBP + LNeP [134]	90.81	87.07
RMP	**91.34**	**88.43**
RCP	**91.80**	**88.03**
CSP	**94.13**	**85.01**
RCSP	**92.91**	**87.12**

4.6.5 Experiments for Ten Expressions

The proposed methods have been implemented with different weights and the results have been tabulated in table 4.17. Among the proposed methods with different weights, RMP method with binary weights achieved an optimal recognition accuracy of 88.43%. The confusion matrix obtained using RMP method with binary weights on ADFES dataset for ten expressions is presented in figure 4.15. The comparison analysis of the proposed methods with the existing variants of binary patterns is reported in the third column of table 4.18. The comparison analysis of proposed method with the existing variants of binary patterns for eight and ten expressions on TFEID and ADFES datasets is shown in figure 4.16. The experimental results from table 4.18 indicate that the proposed methods outperformed the existing variants of binary patterns in terms of recognition accuracy.

4.7 Summary

The main objective of FER systems is to develop feature descriptors that could accurately classify the facial expressions into various categories. Towards realizing this task, texture based feature descriptors namely RMP, RCP, CSP and RCSP have been presented in this Chapter. RMP, a local texture based approach generates two feature codes that are unique to corner, edge and flat regions. RCP, CSP and RCSP feature descriptors have been proposed for overcoming some of the limitations of the existing methods such as CP, LGC and it's variants. In this Chapter, apart from RMP, the experiments are conducted using RCP and CSP independently and with their fusion RCSP by using different weights on a variety of facial expression datasets. From the experimental results, an observation has been made that proposed methods outperformed standard existing methods proving the robustness of the proposed descriptors and in most of the experiments, RCP method has achieved better recognition accuracy than other feature descriptors (RMP, CSP, RCSP). Also, by using different weights to the proposed feature descriptors resulted in an enhanced performance with decreased fv length.

Chapter 5

Graph Structure Inspired Feature Descriptors

Automatic FER is an important research area in computer vision because it has many real-time applications. However, the main issue lies in the design of a feature descriptor that could effectively capture the appearance changes in the facial images. Hence, towards capturing significant features, texture based feature descriptors inspired by the shape of various graphs have been proposed in this Chapter. The main contributions of this Chapter are summarized as follows:

- Novel texture based feature extraction methods named WGFD, inspired by the Windmill graph (Wd(4,2)) have been proposed for extracting the facial features in a local neighborhood. To reduce the fv length of the proposed WGFD methods, the concept of different weights (binary, fibonacci, prime, natural, squares and odd) have been applied.

- PGBP, inspired by the Generalized Petersen Graph (GPG(6,2)) has been proposed for extracting facial features in a local neighborhood. PGBP has been modelled in such a manner that it effectively captures both neighboring pixel and adjacent pixel relationship in a local neighborhood.

- LTrP named Mini Triangular Pattern (mTP) and Mega Triangular Pattern (MTP) have

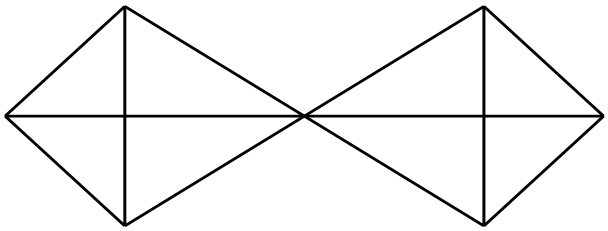


Figure 5.1: Windmill graph, Wd(4,2) with 7 vertices and 12 edges

been proposed for extracting the facial features. LTrP methods have been developed with an intention to minimize the fv length and to maximize the recognition accuracy.

In section 5.1, the feature descriptors inspired from the shape of a Wd(4,2) graph have been proposed. In section 5.2, PGBP method, inspired by the shape of GPG(6,2) graph has been proposed. In section 5.3, LTrP methods, inspired by the shape of a Triangle have been proposed. In section 5.4, the experimental results corresponding to $WGFD_h$, $WGFD_v$, PGBP, mTP and MTP with have been presented and analyzed. In section 5.5, the contributions in this Chapter have been summarized.

5.1 Windmill Graph Inspired Feature Descriptors

5.1.1 Windmill Graph

Windmill graph, Wd(a,b) is an undirected graph, constructed for a$\geq$2 and b$\geq$2 formed by joining 'b' copies of the complete graph (K_a) at a shared universal vertex [170]. In general, Wd(a,b) has ((a-1)*b)+1 vertices and b*a*((a-1)/2) edges, girth 3, radius 1, diameter 2 and vertex connectivity of 1. There are many variants of Windmill graphs available in the literature. For example, Wd(3,b) is known as Friendship graph (F_b), Wd(2,b) is known as Star graph (S_b) and Wd(3,2) is known as Butterfly graph [171]. Among many such variants, the inspiration for feature extraction is drawn from Wd(4,2) and the graph corresponding to Wd(4,2) is shown in figure 5.1.

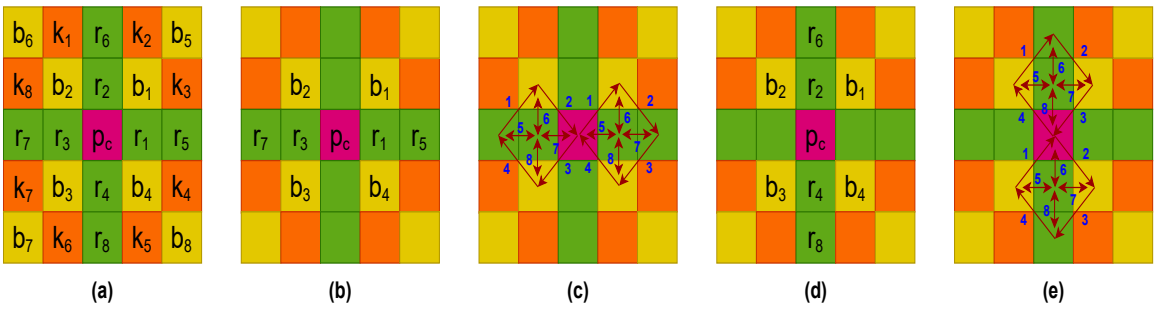


Figure 5.2: Feature extraction process of WGFD methods (a) Logical placement of Rook, Bishop and Knight in a 5 x 5 neighborhood as per RMP (b) Pixels considered for feature extraction using $WGFD_h$ (c) Placement of Wd(4,2) in a 5 x 5 neighborhood, for feature extraction using $WGFD_h$ (d) Pixels considered for feature extraction using $WGFD_v$ (e) Placement of Wd(4,2) in a 5 x 5 neighborhood, for feature extraction using $WGFD_v$

5.1.2 Windmill Graph based Feature Descriptors (WGFD)

Drawing motivation from CP [163], RMP and Wd(4,2), the proposed feature descriptors named WGFD have been modelled. RMP logically fills the 24 pixels surrounding the pixel p_c in a 5 x 5 neighborhood with Rook (r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8), Bishop (b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8) and Knight positions (k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8) as shown in figure 5.2(a). From CP, the concept of placing chessmen in a 5 x 5 neighborhood and from RMP, the numbering scheme for Rook, Bishop and Knight is adopted into the proposed methodology. For the purpose of feature extraction, the inspiration is drawn from the shape of Wd(4,2) graph. Wd(4,2) is logically placed horizontally in a 5 x 5 block for feature extraction using $WGFD_h$ and the pixels considered for feature extraction through $WGFD_h$ are shown in figure 5.2(b). The process of feature extraction using $WGFD_h$ is shown in figure 5.2(c). Similarly, Wd(4,2) is rotated by 90° and is placed vertically in a 5 x 5 block for feature extraction using $WGFD_v$ and the pixels considered for feature extraction through $WGFD_v$ are shown in figure 5.2(d). The process of feature extraction using $WGFD_v$ is shown in figure 5.2(e).

5.1.3 $WGFD_h$ Feature Extraction

For feature extraction using $WGFD_h$, a set of eight pixels named as per RMP are considered, as shown in figure 5.2(b-c). $WGFD_h$ extracts two feature codes in a local neighbor-

hood namely $WGFD_{h_1}$ and $WGFD_{h_2}$. In general image processing applications, the features extracted by the local based approaches majorly focus on capturing the relationship between adjacent pixels or the relationship between the surrounding pixels [17, 57, 134]. Feature extraction using $WGFD_h$ involves capturing the relationship between the adjacent pixels as well as the relationship between the surrounding pixels. For feature extraction of $WGFD_{h_1}$, four pixel positions namely r_7, b_2, p_c and b_3 are considered. Initially, for capturing the adjacent pixels relationship, the pixels r_7, b_2, p_c and b_3 are correspondingly compared with pixels b_2, p_c, b_3 and r_7. In the same manner, for capturing the surrounding pixels relationship, the pixels r_7, b_2, p_c and b_3 are compared sequentially with the pixel r_3. The process of feature extraction using $WGFD_1$ is shown in eq.(5.1). Upon comparison with pixels as shown in eq.(5.1), a sequence of eight bit binary number is obtained. In general image processing applications, binary weights are used for conversion of this binary number to decimal number. Upon using binary weights (shown in eq.(3.27)), the fv length of RCP_1 is 256, as like LBP. To further reduce the fv length of $WGFD_{h_1}$, different weights such as fibonacci (shown in eq.(3.28)) [164], prime (shown in eq.(3.29)), natural (shown in eq.(3.30)), squares (shown in eq.(3.31)) and odd (shown in eq.(3.32)) have been considered. The formulae for feature extraction using $WGFD_{h_1}$ are shown in eq.(5.1) and eq.(5.2).

$$WGFD_1 = \{s(r_7, b_2), s(b_2, p_c), s(p_c, b_3), s(b_3, r_7), s(r_7, r_3), s(b_2, r_3), s(p_c, r_3), s(b_3, r_3)\} \tag{5.1}$$

$$WGFD_{h_1} = \sum (WGFD_1 . * w_m) \tag{5.2}$$

For feature extraction of $WGFD_{h_2}$, four pixel positions namely p_c, b_1, r_5 and b_4 are considered. Initially, for capturing the adjacent pixels relationship, the pixels p_c, b_1, r_5 and b_4 are correspondingly compared with pixels b_1, r_5, b_4 and p_c respectively. In the same manner, for capturing the surrounding pixels relationship, the pixels p_c, b_1, r_5 and b_4 are compared sequentially with the pixel r_1. The process of feature extraction using $WGFD_2$ is shown in eq.(5.3). Upon comparison with pixels as shown in eq.(5.3), a sequence of eight bit binary number is obtained. The formula for feature extraction using $WGFD_{h_2}$ is shown in eq.(5.4). To further reduce the fv length of $WGFD_{h_2}$ also, the concept of different weights have been applied. The histogram features are extracted block wise using $hist_{wgfd_{h_1}}$,

Figure 5.3: (a) Happy expression image from TFEID dataset. Feature response maps generated by (b) $WGFD_{h_1}$ and (c) $WGFD_{h_2}$ using binary weights.

as shown in eq.(5.5) and using $hist_{wgfd_{h_2}}$, as shown in eq.(5.6). Finally, the obtained block wise features ($hist_{wgfd_{h_1}}$, $hist_{wgfd_{h_2}}$) are horizontally concatenated to obtain the fv of $WGFD_h$ feature descriptor, as shown in eq.(5.7). The formula for calculating size of $WGFD_h$ feature descriptor for an input image of size N x N is shown in eq.(5.8). The algorithm for feature extraction through $WGFD_h$ is mentioned in algorithm 5.1. The value of 'L' in algorithm 5.1 and in eq.(5.8) depends on the weight matrix used in the process of feature extraction. For an input image of size N x N, the computational complexity of the proposed $WGFD_h$ method is $O(N^2)$. The feature response maps generated by $WGFD_h$ method using binary weights is shown in figure 5.3.

$$WGFD_2 = \{s(p_c, b_1), s(b_1, r_5), s(r_5, b_4), s(b_4, p_c), s(p_c, r_1), s(b_1, r_1), s(r_5, r_1), s(b_4, r_1)\} \tag{5.3}$$

$$WGFD_{h_2} = \sum (WGFD_1. * w_m) \tag{5.4}$$

$$hist_{wgfd_{h_1}} = Hist(WGFD_{h_1}) \tag{5.5}$$

$$hist_{wgfd_{h_2}} = Hist(WGFD_{h_2}) \tag{5.6}$$

$$fv_{wgfd_h} = hist_{wgfd_{h_1}} \cup hist_{wgfd_{h_2}} \tag{5.7}$$

$$Size(fv_{wgfd_h}) = \lceil (N-4)/C \rceil * \lceil (N-4)/C \rceil * 2 * L \tag{5.8}$$

5.1.4 $WGFD_v$ Feature Extraction

Feature extraction using $WGFD_v$ involves capturing the relationship between the adjacent pixels as well as the relationship between the surrounding pixels. $WGFD_v$ extracts two feature codes in a local neighborhood namely $WGFD_{v_1}$ and $WGFD_{v_2}$. For feature extrac-

Algorithm 5.1 Feature Extraction through $WGFD_h$

Input: An Input image (Img) of size N x N

Output: Feature vector (fv_{wgfd_h}) of size ⌈(N-4)/C⌉ * ⌈(N-4)/C⌉ * 2 * L

procedure $WGFD_H$(Img)
  Initialization: $WGFD_{h_1}$, $WGFD_{h_2}$, $WGFD_h \leftarrow \{\ \}$
  Load an input image (Img)
  for all a ∈ range(1,N-4) **do**
    for all b ∈ range(1,N-4) **do**
      Block = img(a:a+4,b:b+4)
      Assign the pixel values to Rook, Bishop and Knight in the 5 x 5 block.
      Calculate $WGFD_{h_1}$ using eqs.(5.1) and (5.2)
      Calculate $WGFD_{h_2}$ using eqs.(5.3) and (5.4)
    end for
  end for
  Create two feature response maps obtained from $WGFD_{h_1}$ and $WGFD_{h_2}$ by reshaping them to (N-4) x (N-4).
  Each feature response map is partitioned into C x C non-overlapping blocks.
  Histograms are extracted block wise for both the feature response maps using eqs.(5.5) and (5.6).
  Concatenate all the histograms to obtain fv_{wgfd_h} using eq.(5.7).
  return fv_{wgfd_h}
end procedure

tion of $WGFD_{v_1}$, four pixel positions namely b_2, r_6, b_1 and p_c are considered. Initially, for capturing the adjacent pixels relationship, the pixels b_2, r_6, b_1 and p_c are correspondingly compared with pixels r_6, b_1, p_c and b_2 respectively. In the same manner, for capturing the surrounding pixels relationship, the pixels b_2, r_6, b_1 and p_c are compared sequentially with the pixel r_2. The process of feature extraction using $WGFD_3$ is shown in eq.(5.9). Upon comparison with pixels as shown in eq.(5.9), a sequence of eight bit binary number is obtained. The formulae for feature extraction using $WGFD_{v_1}$ are shown in eqs.(5.9) and (5.10). To further reduce the fv length of $WGFD_{v_1}$ also, the concept of different weights have been applied.

$$WGFD_3 = \{s(b_2, r_6), s(r_6, b_1), s(b_1, p_c), s(p_c, b_2), s(b_2, r_2), s(r_6, r_2), s(b_1, r_2), s(p_c, r_2)\} \tag{5.9}$$

$$WGFD_{v_1} = \sum (WGFD_1. * w_m) \tag{5.10}$$

For feature extraction of $WGFD_{v_2}$, four pixel positions namely b_3, p_c, b_4 and r_8 are considered. Initially, for capturing the adjacent pixels relationship, the pixels b_3, p_c, b_4 and

r_8 are correspondingly compared with pixels p_c, b_4, r_8 and b_3 respectively. In the same manner, for capturing the surrounding pixels relationship, the pixels b_3, p_c, b_4 and r_8 are compared sequentially with the pixel r_4 respectively. The process of feature extraction using $WGFD_4$ is shown in eq.(5.11). Upon comparison with pixels as shown in eq.(5.11), a sequence of eight bit binary number is obtained. The formulae for feature extraction using $WGFD_{v_2}$ are shown in eqs.(5.11) and (5.12). To further reduce the fv length of $WGFD_{v_2}$ also, the concept of different weights have been applied. The histogram features are extracted block wise using $WGFD_{v_1}$, as shown in eq.(5.13) and using $WGFD_{v_2}$, as shown in eq.(5.14). Finally, the obtained block wise features ($hist_{wgfd_{v_1}}$, $hist_{wgfd_{v_2}}$) are horizontally concatenated to obtain fv_{wgfd_v}, as shown in eq.(5.15). The formula for calculating size of $WGFD_v$ feature descriptor for an input image of size N x N is shown in eq.(5.16). The algorithm for feature extraction through $WGFD_v$ is mentioned in algorithm 5.2. The value of 'L' in algorithm 5.2 and in eq.(5.16) depends on the weight matrix used in the process of feature extraction. For an input image of size N x N, the computational complexity of the proposed $WGFD_v$ method is $O(N^2)$. The feature response maps generated by $WGFD_v$ method using binary weights is shown in figure 5.4. Thus, in feature extraction of WGFD methods, both the relationship between the adjacent pixels and the surrounding pixels have been considered for extracting robust and discriminative information in a local neighborhood.

$$WGFD_4 = \{s(b_3, p_c), s(p_c, b_4), s(b_4, r_8), s(r_8, b_3), s(b_3, r_4), s(p_c, r_4), s(b_4, r_4), s(r_8, r_4)\} \tag{5.11}$$

$$WGFD_{v_2} = \sum (WGFD_1 . * w_m) \tag{5.12}$$

$$hist_{wgfd_{v_1}} = Hist(WGFD_{v_1}) \tag{5.13}$$

$$hist_{wgfd_{v_2}} = Hist(WGFD_{v_2}) \tag{5.14}$$

$$fv_{wgfd_v} = hist_{wgfd_{v_1}} \cup hist_{wgfd_{v_2}} \tag{5.15}$$

$$Size(fv_{wgfd_v}) = \lceil (N-4)/C \rceil * \lceil (N-4)/C \rceil * 2 * L \tag{5.16}$$

Algorithm 5.2 Feature Extraction through $WGFD_v$

Input: An Input image (Img) of size N x N
Output: Feature vector (fv_{wgfd_v}) of size ⌈(N-4)/C⌉ * ⌈(N-4)/C⌉ * 2 * L

procedure $WGFD_v$(Img)
  Initialization: $WGFD_{v_1}$, $WGFD_{v_2}$, $WGFD_v \leftarrow \{\ \}$
  Load an input image (Img)
  for all a ∈ range(1,N-4) **do**
    for all b ∈ range(1,N-4) **do**
      Block = img(a:a+4,b:b+4)
      Assign the pixel values to Rook, Bishop and Knight in the 5 x 5 block.
      Calculate $WGFD_{v_1}$ using eqs.(5.9) and (5.10)
      Calculate $WGFD_{v_2}$ using eqs.(5.11) and (5.12)
    end for
  end for
  Create two feature response maps obtained from $WGFD_{v_1}$ and $WGFD_{v_2}$ by reshaping them to (N-4) x (N-4).
  Each feature response map is partitioned into C x C non-overlapping blocks.
  Histograms are extracted block wise for both the feature response maps using eqs.(5.13) and (5.14).
  Concatenate all the histograms to obtain fv_{wgfd_v} using eq.(5.15).
  return fv_{wgfd_v}
end procedure

5.2 Petersen Graph Inspired Feature Descriptor

5.2.1 Generalized Petersen Graph

"Generalized Petersen Graph (GPG) is a connected cubic graph formed by connecting the vertices of a regular polygon to the corresponding vertices of a star polygon" [172]. They include the Petersen graph and generalize one of the ways of constructing the Petersen

Figure 5.4: (a) Happy expression image from TFEID dataset. Feature response maps generated by (b) $WGFD_{v_1}$ and (c) $WGFD_{v_2}$ using binary weights.

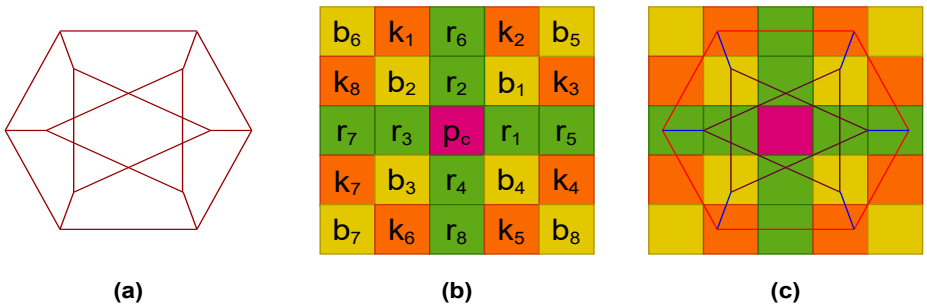


Figure 5.5: (a) Generalized Petersen Graph, GPG(6,2) (b) Sample 5 x 5 block with numbering given for pixels for feature extraction through PGBP (c) Feature extraction through PGBP

graph. GPG is shown in figure 5.5(a). Drawing inspiration from GPG(6,2), Petersen Graph based Binary Pattern (PGBP), a graph based feature extraction technique has been proposed for extracting facial features in a local 5 x 5 neighborhood. In figure 5.5(b), numbering is given to pixels in a 5 x 5 block. In figure 5.5(c), GPG(6,2) is placed in a 5 x 5 block for the purpose of feature extraction.

5.2.2 Petersen Graph based Binary Pattern (PGBP) Feature Extraction

PGBP extracts three features from a 5 x 5 overlapping neighborhood. As seen from figure 5.5(c), there are eighteen edges in the generalized petersen graph. These 18 edges are logically separated into three groups. The first group contains the edges (represented in figure 5.5(c) using pink color) that are part of the inner star polygon. The second group contains the edges (represented in figure 5.5(c) using blue color) that are connecting the inner star polygon to outer regular polygon. The third group contains the edges (represented in figure 5.5(c) using red color) that are part of the outer regular polygon. These three groups are a basis for feature extraction through PGBP. Corresponding to three groups, three features namely $PGBP_1$, $PGBP_2$ and $PGBP_3$ are extracted.

5.2.2.1 $PGBP_1$

For feature extraction through $PGBP_1$, the vertices of the inner star polygon are considered (r_1, b_2, b_3, r_3, b_4, b_1). Initially, starting from the pixel r_1, the triangular vertices (b_2, b_3) are

compared in a counter clockwise manner with the pixel p_c. Next, starting from the pixel r_3, the triangular vertices (b_4, b_1) are compared in a counter clockwise manner with the pixel p_c. Thus, upon performing these operations, a six bit binary number is obtained which is then converted into a decimal number. The corresponding equations for feature extraction through $PGBP_1$ are shown in eqs.(5.17) to (5.19). As, there are only six bits involved in the fv of $PGBP_1$, only the concept of binary weights has been used for feature extraction using PGBP. The equation corresponding to binary weights considered for PGBP is shown in eq.(5.19).

$$PGBP_a = \{s(r_1, p_c), s(b_2, p_c), s(b_3, p_c), s(r_3, p_c), s(b_4, p_c), s(b_1, p_c)\} \tag{5.17}$$

$$PGBP_1 = \sum (PGBP_a. * W) \tag{5.18}$$

$$W = \Big[1, \quad 2, \quad 4, \quad 8, \quad 16, \quad 32\Big] \tag{5.19}$$

5.2.2.2 $PGBP_2$

For feature extraction through $PGBP_2$, the vertices of the inner star polygon (r_1, b_2, b_3, r_3, b_1, b_4) and the vertices of outer regular polygon (r_5, k_2, k_1, r_7, k_6, k_5) are considered. Initially, starting from the pixel r_1, the corresponding vertices of star polygon are compared with the vertices of outer regular polygon as shown in figure 5.5(c). The sequence of pixels chosen for comparison follows counter clockwise manner. Thus, for $PGBP_2$, the pixels corresponding to both polygons are considered. Upon performing these operations, a six bit binary number is obtained which is then converted into a decimal number. The corresponding equations for feature extraction through $PGBP_2$ are shown in eqs.(5.20) and (5.21).

$$PGBP_b = \{s(r_1, r_5), s(b_1, k_2), s(b_2, k_1), s(r_3, r_7), s(b_3, k_6), s(b_4, k_5)\} \tag{5.20}$$

$$PGBP_2 = \sum (PGBP_b. * W) \tag{5.21}$$

5.2.2.3 $PGBP_3$

For feature extraction through $PGBP_3$, the vertices of outer regular polygon (k_1, k_2, r_5, k_5, k_6, r_7) are considered. Initially, starting from the pixel k_1, the corresponding vertices of outer regular polygon are sequentially compared as shown in eq.(5.22) and in figure 5.5(c).

Figure 5.6: (a) Happy expression image from TFEID dataset. Feature response maps generated by (b) $PGBP_1$ (c) $PGBP_1$ and (d) $PGBP_3$

The sequence of pixels chosen for comparison analysis follows clockwise manner. Upon performing these operations, a six bit binary number is obtained which is then converted into a decimal number. The corresponding equations for feature extraction through $PGBP_3$ are shown in eqs.(5.22) and (5.23).

$$PGBP_c = \{s(k_1, k_2), s(k_2, r_5), s(r_5, k_5), s(k_5, k_6), s(k_6, r_7), s(r_7, k_1)\} \tag{5.22}$$

$$PGBP_3 = \sum(PGB_{Pc.} * W) \tag{5.23}$$

5.2.2.4 Feature Level Fusion

Literature studies suggest that feature level fusion and extracting block wise features increases recognition accuracy [17]. Hence, the block wise histograms are calculated for all the three features, as shown in eqs.(5.24) to (5.26). $hist_{pgbp_1}$, $hist_{pgbp_2}$ and $hist_{pgbp_3}$ correspond to the block wise histograms of $PGBP_1$, $PGBP_2$ and $PGBP_3$ respectively. Finally, all the obtained block wise histogram features from $PGBP_1$, $PGBP_2$ and $PGBP_3$ are horizontally concatenated to obtain the feature vector corresponding to PGBP (fv_{pgbp}), as shown in eq.(5.27).

$$hist_{pgbp_1} = Hist(PGBP_1) \tag{5.24}$$

$$hist_{pgbp_2} = Hist(PGBP_2) \tag{5.25}$$

$$hist_{pgbp_3} = Hist(PGBP_3) \tag{5.26}$$

$$fv_{pgbp} = hist_{pgbp_1} \cup hist_{pgbp_2} \cup hist_{pgbp_3} \tag{5.27}$$

The algorithm for feature extraction through PGBP is mentioned in algorithm 5.3. The value of 'L' in algorithm 5.3 is 64. For an input image of size N x N, the computational complexity of the proposed PGBP method is $O(N^2)$. The feature response maps generated

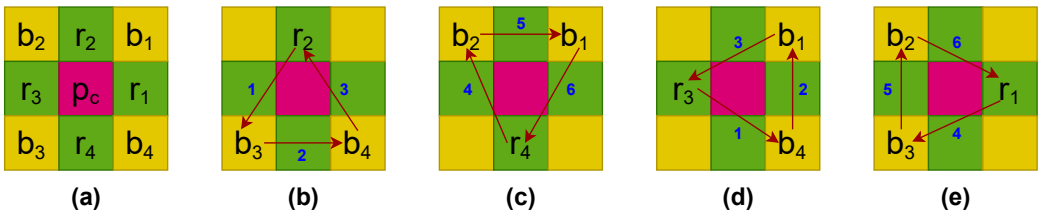


Figure 5.7: Procedure for extracting features through mTP (a-b) Feature extraction using mTP_1 (c-d) Feature extraction using mTP_2

by PGBP method using binary weights is shown in figure 5.6. Thus, in the feature extraction of PGBP method, the relationship between the adjacent pixels, relationship with the surrounding pixels and also the multi-distance relationship between the pixels have been considered for extracting robust and discriminative information in a local neighborhood.

Algorithm 5.3 Feature Extraction through PGBP

Input: An Input image (Img) of size N x N
Output: Feature vector (fv_{pgbp}) of size size $\lceil$(N-4)/C$\rceil$ * $\lceil$(N-4)/C$\rceil$ * 3 * L

procedure PGBP(Img)
  Initialization: $PGBP_1$, $PGBP_2$, $PGBP_3$, PGBP $\leftarrow$ { }
  Load an input image (Img)
  for all a $\in$ range(1,N-4) **do**
    for all b $\in$ range(1,N-4) **do**
      Block = img(a:a+4,b:b+4)
      Assign the pixel values to Rook, Bishop and Knight in the 5 x 5 block.
      Calculate $PGBP_1$ using eqs.(5.17) to (5.19)
      Calculate $PGBP_2$ using eqs.(5.20) and (5.21)
      Calculate $PGBP_3$ using eqs.(5.22) and (5.23)
    end for
  end for
  Create three feature response maps obtained from $PGBP_1$, $PGBP_2$ and $PGBP_3$ by reshaping each of them to (N-4) x (N-4).
  Each feature response map is partitioned into C x C non-overlapping blocks.
  Histograms are extracted block wise from all the three feature response maps using eqs.(5.24) to (5.26).
  Concatenate all the histograms to obtain fv_{pgbp} using eq.(5.27).
  return fv_{pgbp}
end procedure

5.3 Local Triangular Patterns (LTrP)

In this section, texture based feature descriptors inspired by the shape of a Triangle have been proposed. Towards extracting significant features, Local Triangular Patterns (LTrP) named mini Triangular Pattern (mTP) and Mega Triangular Pattern (MTP) have been proposed for extracting the facial features in 3 x 3 and 5 x 5 neighborhoods respectively. The proposed methods (mTP and MTP) have been inspired from CP [163] and RMP. The numbering scheme followed for extracting features through mTP is shown in figure 5.7(a). The pixels in a 3 x 3 neighborhood are logically named as per the methodology employed in kTP. The numbering scheme of chessmen as per RMP in a 5 x 5 neighborhood is shown in figure 5.8(a) and that of MTP is shown in figure 5.8(b). From a 5 x 5 block, for MTP, only the pixels corresponding to circular neighborhood of radius 2 (rd = 2) are considered. Thus, mTP considers 3 x 3 circular neighborhood (rd = 1) for feature extraction, whereas, MTP considers 5 x 5 circular neighborhood (rd = 2) for feature extraction. The procedure of feature extraction through mTP is discussed in section 5.3.1 and through MTP is discussed in section section 5.3.2.

5.3.1 Mini Triangular Pattern (mTP) Feature extraction

Inspired by CP, chessmen have been employed in a 3 x 3 neighborhood, as shown in figure 5.7(a). In a 3 x 3 neighborhood (rd = 1), the 4-neighbors logically constitute to Rook, whereas diagonal neighbors constitute to Bishop. In figure 5.7(a), numbering is given to chessmen such as Rook (r_1, r_2, r_3, r_4) and Bishop (b_1, b_2, b_3, b_4). The numbering scheme follows counter clockwise direction for extracting features. For the purpose of feature extraction, the emphasis is placed on adjacent pixels, rather than comparison with the pixel p_c. This is achieved by forming four triangles (two vertical and two horizontal) with reference to four Rook positions (r_1, r_2, r_3, r_4). The vertical triangles are formed corresponding to Rook positions (r_2 and r_4) and the horizontal triangles are formed corresponding to Rook positions (r_3 and r_1).

Initially, the position r_2 is chosen and is connected with two Bishop positions (b_3, b_4)

to form a triangle in vertical direction. Rather than comparing each of these pixel positions with the pixel p_c, the subsequent pixels are compared in counter clockwise manner starting from the Rook position (r_2), as shown in figure 5.7(b). Next, the position r_4 is chosen and is connected with two Bishop positions (b_2, b_1) to form another triangle in vertical direction and the subsequent pixels are compared in a clockwise manner, as shown in figure 5.7(c). Thus, by combining these two, mTP_a is obtained, which is shown in eq.(5.28). After comparing with subsequent pixels in a manner as shown in eq.(5.28), six binary bits are obtained which are then multiplied by W to obtain fv for mTP_1. The corresponding equations for complete feature extraction through mTP_1 are shown in eqs.(5.28) and (5.29).

Similarly, the triangles can also be formed in horizontal direction by considering other two Rook positions (r_3, r_1). So, the position r_3 is chosen and is connected and compared with two Bishop positions (b_4, b_1) in counter clockwise manner, as shown in figure 5.7(d). Next, the position r_1 is chosen and is connected with two Bishop positions (b_3, b_2) and the subsequent pixels are compared in a clockwise manner, as shown in figure 5.7(d). Thus, by combining these two, mTP_b is obtained, which contains six binary bits. The six binary bits are then multiplied by W to obtain fv for mTP_2. The corresponding equations for feature extraction through mTP_2 are shown in eqs.(5.30) and (5.31). The block wise histograms are calculated for all the two features, as shown in eqs.(5.32) and (5.33). $hist_{mTP_1}$ and $hist_{mTP_2}$ correspond to the block wise histograms of mTP_1 and mTP_2 respectively. Finally, all the obtained block wise histogram features from mTP_1 and mTP_2 are horizontally concatenated to obtain fv_{mTP}, as shown in eq.(5.34). The algorithm for feature extraction through mTP is mentioned in algorithm 5.4. The value of 'L' in algorithm 5.4 is 64. For an input image of size N x N, the computational complexity of the proposed mTP method is $O(N^2)$. The feature response maps generated by mTP method are shown in figure 5.9.

$$mTP_a = \{s(r_2, b_3), s(b_3, b_4), s(b_4, r_2), s(r_4, b_2), s(b_2, b_1), s(b_1, r_4)\} \tag{5.28}$$

$$mTP_1 = \sum (mTP_a. * W) \tag{5.29}$$

$$mTP_b = \{s(r_3, b_4), s(b_4, b_1), s(b_1, r_3), s(r_1, b_3), s(b_3, b_2), s(b_2, r_1)\} \tag{5.30}$$

$$mTP_2 = \sum (mTP_b. * W) \tag{5.31}$$

$$hist_{mTP_1} = Hist(mTP_1) \tag{5.32}$$

$$hist_{mTP_2} = Hist(mTP_2) \tag{5.33}$$

$$fv_{mTP} = hist_{mTP_1} \cup hist_{mTP_2} \tag{5.34}$$

Algorithm 5.4 Feature Extraction through mTP

Input: An Input image (Img) of size N x N
Output: Feature vector (fv_{mtp}) of size size $\lceil$(N-2)/C$\rceil$ * $\lceil$(N-2)/C$\rceil$ * 2 * L

procedure MTP(Img)
  Initialization: mTP_1, mTP_2, mTP $\leftarrow$ { }
  Load an input image (Img)
  for all a $\in$ range(1,N-2) **do**
    for all b $\in$ range(1,N-2) **do**
      Block = img(a:a+2,b:b+2)
      Assign the pixel values to Rook and Bishop in the 3 x 3 block.
      Calculate mTP_1 using eqs.(5.28) and (5.29)
      Calculate mTP_2 using eqs.(5.30) and (5.31)
    end for
  end for
  Create two feature response maps obtained from mTP_1 and mTP_2 by reshaping each of them to (N-2) x (N-2).
  Each feature response map is partitioned into C x C non-overlapping blocks.
  Histograms are extracted block wise from all the two feature response maps using eqs.(5.32) and (5.33).
  Concatenate all the histograms to obtain fv_{mTP} using eq.(5.34).
  return fv_{mTP}
end procedure

5.3.2 Mega Triangular Pattern (MTP) Feature extraction

In CP, all the pixels in a 5 x 5 neighborhood are considered for feature extraction and CP extracts six features, which leads to high fv length. In order to maximize recognition accuracy and to minimize fv length, MTP is proposed. The fv length of MTP is 128. Also, MTP considers only twelve surrounding pixels in a 5 x 5 neighborhood rather than twenty four surrounding pixels as like CP. The pixels considered and the numbering scheme for feature extraction through RMP and MTP are shown in figure 5.8(a-b). From figure 5.8(b), out of twelve neighboring pixels, eight pixels correspond to Knight positions (k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8) and four pixels correspond to Rook positions (r_1, r_2, r_3, r_4). So, corresponding to four Rook positions from figure 5.8(c-f), four triangles (2 vertical and 2 horizontal) are

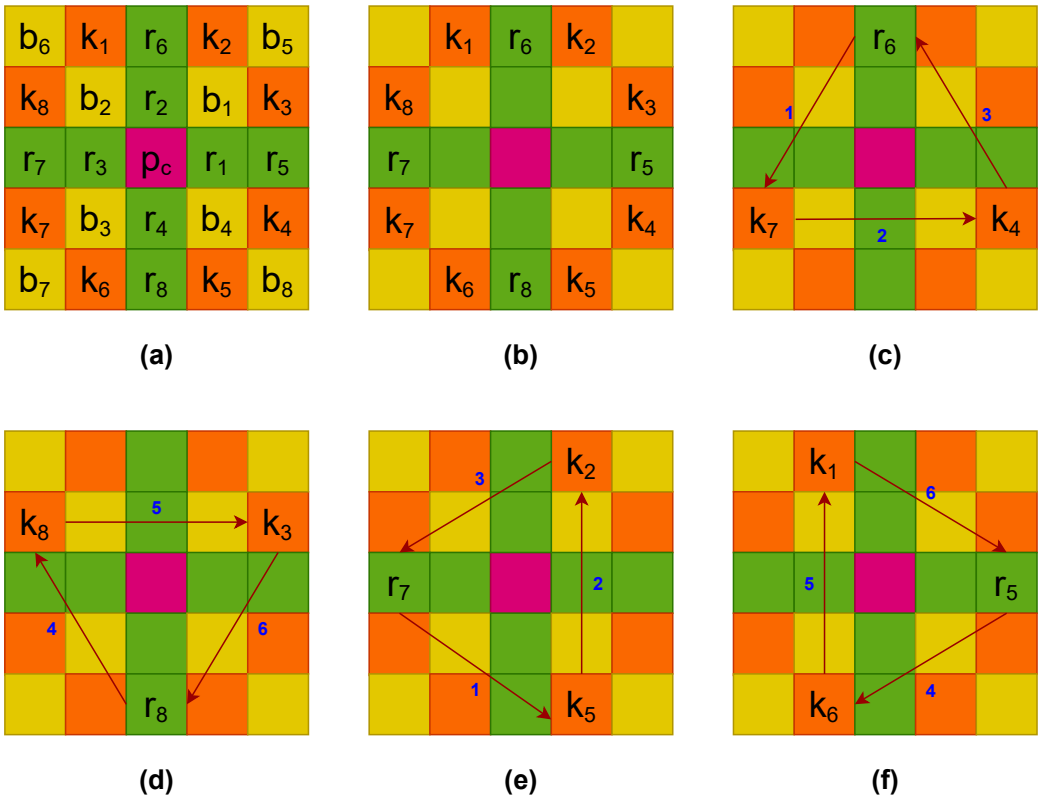


Figure 5.8: Procedure for extracting features through MTP (a-b) Feature extraction using MTP_1 (c-d) Feature extraction using MTP_2

Figure 5.9: (a) Happy expression image from TFEID dataset. Feature response maps generated by (b) mTP_1 and (c) mTP_2

(a) (b) (c)

Figure 5.10: (a) Happy expression image from TFEID dataset. Feature response maps generated by (b) MTP_1 and (c) MTP_2

formed. Initially, the position r_6 is chosen and is connected with two Knight positions (k_7, k_4). Rather than comparing each of these pixel positions with the pixel p_c, the emphasis is placed on comparing subsequent pixels in counter clockwise manner, connected in the form of a triangle, as shown in figure 5.8(c). Next, the position r_8 is chosen and is connected with two Knight positions (k_8, k_3) and the subsequent pixels are compared in a clockwise manner, as shown in figure 5.8(d). Thus, by combining these two, MTP_a is obtained, which contains six binary bits. These six binary bits are then multiplied by W to obtain fv of MTP_1. The corresponding equations for feature extraction through MTP_1 are shown in eqs. (5.35) and (5.36).

Similarly, the triangles can be formed in horizontal directions also. So, the position r_7 is chosen and is connected with two Knight positions (k_5, k_2) and the subsequent pixels are compared in counter clockwise manner, connected in the form of a triangle, as shown in figure 5.8(e). Next, the position r_5 is chosen and is connected with two Bishop positions (k_6, k_1) and the subsequent pixels are compared in clockwise manner, connected in the form of a triangle, as shown in figure 5.8(f). Thus, by combining these two, MTP_b is obtained, which contains six binary bits. These six binary bits are then multiplied by W to obtain feature vector MTP_2. The corresponding equations for feature extraction through MTP_2 are shown in eqs.(5.37) and (5.38). The block wise histograms are calculated for all the two features, as shown in eqs.(5.39) and (5.40). $hist_{MTP_1}$ and $hist_{MTP_2}$ correspond to the block wise histograms of MTP_1 and MTP_2 respectively. Finally, all the obtained block wise histogram features from MTP_1 and MTP_2 are horizontally concatenated to obtain fv_{MTP}, as shown in eq.(5.41). The algorithm for feature extraction through mTP is mentioned in algorithm 5.5. For an input image of size N x N, the computational complexity of the

proposed MTP method is O(N^2). The feature response maps generated by MTP method are shown in figure 5.10.

$$MTP_a = \{s(r_6, k_7), s(k_7, k_4), s(k_4, r_6), s(r_8, k_8), s(k_8, k_3), s(k_3, r_8)\} \tag{5.35}$$

$$MTP_1 = \sum(MTP_a. * W) \tag{5.36}$$

$$MTP_b = \{s(r_7, k_5), s(k_5, k_2), s(k_2, r_7), s(r_5, k_6), s(k_6, k_1), s(k_1, r_5)\} \tag{5.37}$$

$$MTP_2 = \sum(MTP_b. * W) \tag{5.38}$$

$$hist_{MTP_1} = Hist(MTP_1) \tag{5.39}$$

$$hist_{MTP_2} = Hist(MTP_2) \tag{5.40}$$

$$fv_{MTP} = hist_{MTP_1} \cup hist_{MTP_2} \tag{5.41}$$

Algorithm 5.5 Feature Extraction through MTP

Input: An Input image (Img) of size N x N
Output: Feature vector (fv_{MTP}) of size size $\lceil$(N-4)/C$\rceil$ * $\lceil$(N-4)/C$\rceil$ * 2 * L

procedure MTP(Img)
  Initialization: MTP_1, MTP_2, MTP $\leftarrow$ { }
  Load an input image (Img)
  for all a $\in$ range(1,N-4) **do**
    for all b $\in$ range(1,N-4) **do**
      Block = img(a:a+4,b:b+4)
      Assign the pixel values to Rook and Knight in the 5 x 5 block.
      Calculate MTP_1 using eqs.(5.35) and (5.36)
      Calculate MTP_2 using eqs.(5.37) and (5.38)
    end for
  end for
  Create two feature response maps obtained from MTP_1 and MTP_2 by reshaping each of them to (N-4) x (N-4).
  Each feature response map is partitioned into C x C non-overlapping blocks.
  Histograms are extracted block wise from all the two feature response maps using eqs.(5.39) and (5.40).
  Concatenate all the histograms to obtain fv_{MTP} using eq.(5.41).
  return fv_{MTP}
end procedure

5.4 Results and Comparison Analysis

For experimental analysis, multi-class SVM employing One vs One (OVO) and One vs All (OVA) approaches with a linear kernel is followed. In this section, the fv length comparison of proposed methods, the experimental results and the comparison of proposed methods with the existing methods is reported.

5.4.1 Feature Vectors Comparison

As WGFD methods generate two features each, the fv length of WGFD methods is same as that of RMP and RCP. Thus, to the proposed feature descriptors, different weights have been applied to effectively reduce the fv length and to find out the optimal recognition accuracy. PGBP method generates three feature codes, each of length six bits. Hence, the fv length corresponding to PGBP is 192. LTrP (mTP and MTP) methods generate two feature codes, each of length six bits. Hence, the fv length corresponding to mTP and MTP methods is 128.

5.4.2 Experiments for Six Expressions

The experiments for six expressions have been conducted on different 'in the lab' datasets. The proposed WGFD methods have been implemented with different weights and the results have been tabulated. In table 5.1, for each dataset, the recognition accuracy comparison of $WGFD_h$ with different weights using OVO-SVM classifier, in table 5.2, the recognition accuracy comparison of $WGFD_h$ with different weights using OVA-SVM classifier is shown. In table 5.3, the recognition accuracy comparison of $WGFD_v$ with different weights using OVO-SVM classifier and in table 5.4, the recognition accuracy comparison of $WGFD_v$ with different weights using OVA-SVM classifier is shown. In tables 5.5, 5.6 and 5.7 the recognition accuracy of PGBP, mTP and MTP with OVO-SVM and OVA-SVM classifiers for different 'in the lab' datasets is shown. Among the proposed methods with different weights, $WGFD_v$ method with fibonacci, binary / prime and prime weights achieved an optimal recognition accuracy of 67.58%, 91.67% and 94.70% on

Table 5.1: Recognition accuracy of $WGFD_h$ with different weights for six expressions using OVO-SVM classifier for different 'in the lab' datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	59.97	61.61	61.67	61.61	61.64	60.58
MUG	86.88	86.81	87.05	86.22	86.81	86.30
CK+	89.93	90.81	90.71	91.67	90.59	91.79
OULU	75.21	76.53	76.67	75.97	77.01	76.39
TFEID	95.08	95.08	95	94.17	95.5	95.42
KDEF	83.57	83.33	83.57	82.62	83.33	82.86
WSEFEP	87.78	85.56	88.33	87.78	87.78	87.78
ADFES	90.91	91.67	92.42	91.67	93.18	89.39

Table 5.2: Recognition accuracy of $WGFD_h$ with different weights for six expressions using OVA-SVM classifier for different 'in the lab' datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	60.70	63.86	62.19	62.72	65.06	62.78
MUG	88.30	87.93	87.55	87.19	87.85	87.70
CK+	90.25	92.31	91.99	92.53	90.9	91.99
OULU	76.46	75.69	76.53	74.24	77.08	74.44
TFEID	94.58	95.08	94.17	93.63	94.67	94.46
KDEF	83.57	84.52	83.33	83.57	83.10	81.19
WSEFEP	90	89.44	91.11	90.56	88.89	89.44
ADFES	93.94	93.94	93.18	91.67	93.94	93.94

JAFFE, WSEFEP and ADFES datasets respectively. $WGFD_v$ method with squares weights achieved an optimal recognition accuracy of 78.12% and 96.07% on OULU and TFEID datasets respectively. The proposed mTP method achieved an optimal recognition accuracy of 89.26% and 92.95% on MUG and CK+ datasets respectively. MTP and mTP methods achieved an optimal recognition accuracy of 84.52% on KDEF dataset.

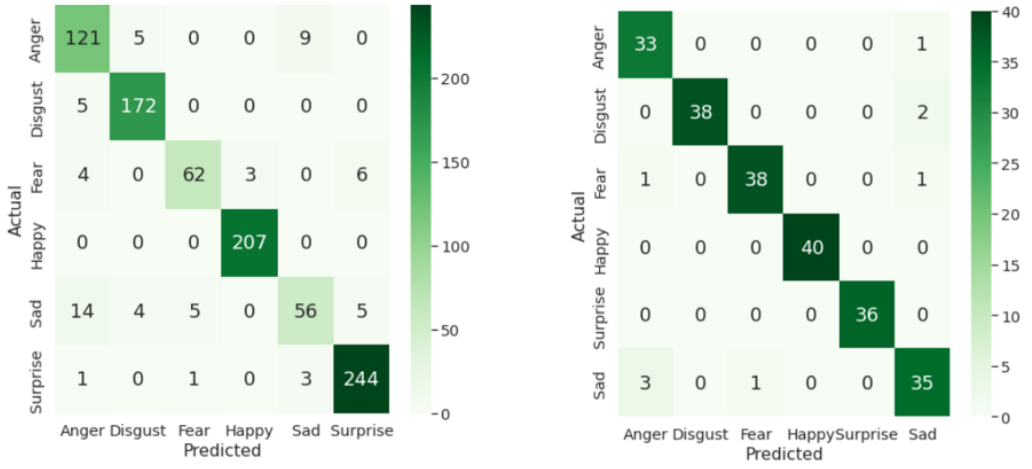
The confusion matrix obtained using mTP method for CK+ dataset is presented in figure 5.11(a) and for TFEID dataset using $WGFD_v$ method with squares weights is presented in figure 5.11(b). The confusion matrix obtained using MTP method for KDEF dataset is presented in figure 5.12(a) and for ADFES dataset using $WGFD_v$ method with prime weights is presented in figure 5.12(b). In table 5.8, the comparison analysis of the proposed methods with the existing variants of binary patterns, implemented in our environment setup is shown. In table 5.9, the comparison analysis of the proposed methods with the existing

Table 5.3: Recognition accuracy of $WGFD_v$ with different weights for six expressions using OVO-SVM classifier for different 'in the lab' datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	62.20	62.23	59.38	61.08	63.41	60.09
MUG	86.37	86	86.37	85.11	85.85	85.85
CK+	90.37	91.24	91.13	91.56	90.91	91.89
OULU	75.56	77.71	78.06	76.67	78.12	76.94
TFEID	94.17	93.67	94.58	94.17	95.08	94.58
KDEF	84.05	84.29	83.10	82.62	84.05	82.86
WSEFEP	87.22	88.33	87.22	88.89	86.11	86.67
ADFES	91.67	90.91	91.67	89.39	90.15	88.64

Table 5.4: Recognition accuracy of $WGFD_v$ with different weights for six expressions using OVA-SVM classifier for different 'in the lab' datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	64.48	67.58	64.33	64.51	66.68	62.84
MUG	87.33	86.89	87.26	86.59	87.48	87.48
CK+	90.24	91.76	91.77	92.21	90.68	91.98
OULU	76.64	76.67	76.60	75.42	76.74	75.90
TFEID	95.42	94.17	94.38	94.04	96.07	94.46
KDEF	83.57	83.57	83.33	84.52	84.29	83.10
WSEFEP	91.67	88.89	91.67	88.33	90	91.11
ADFES	93.18	93.18	94.70	92.42	93.18	93.18



(a) (b)

Figure 5.11: Confusion matrix for six expressions on (a) CK+ dataset using mTP method (b) TFEID dataset using $WGFD_v$ method with squares weights

Table 5.5: Recognition accuracy of PGBP for six expressions for different 'in the lab' datasets

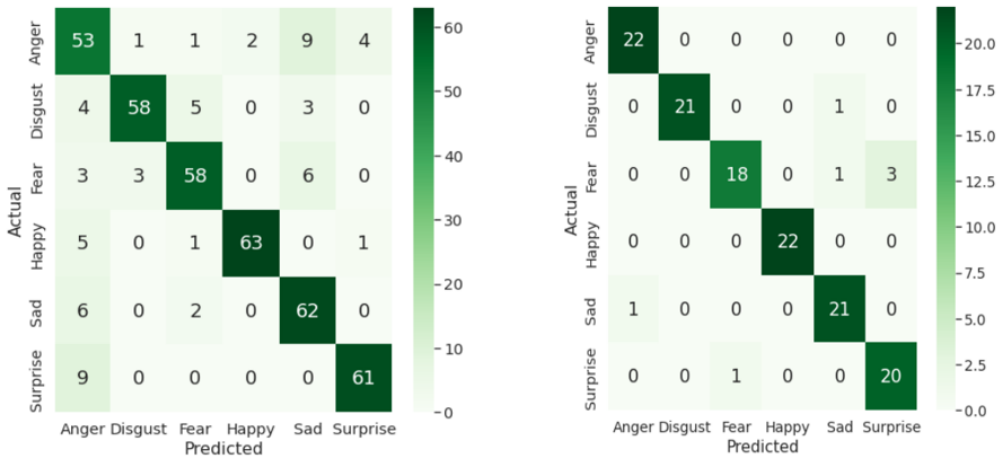
Classifier	**JAFFE**	**MUG**	**CK+**	**OULU**	**TFEID**	**KDEF**	**WSEFEP**	**ADFES**
OVO-SVM	58.88	87.85	91.66	74.38	94.32	84.05	87.22	93.18
OVA-SVM	62.75	88.89	91.55	76.32	95.29	84.52	90	93.18

Table 5.6: Recognition accuracy of mTP for six expressions for different 'in the lab' datasets

Classifier	**JAFFE**	**MUG**	**CK+**	**OULU**	**TFEID**	**KDEF**	**WSEFEP**	**ADFES**
OVO-SVM	59.65	87.78	92.95	74.31	95	82.86	87.78	90.91
OVA-SVM	62.87	89.26	92.51	76.39	94.50	84.52	90	93.94

Table 5.7: Recognition accuracy of MTP for six expressions for different 'in the lab' datasets

Classifier	**JAFFE**	**MUG**	**CK+**	**OULU**	**TFEID**	**KDEF**	**WSEFEP**	**ADFES**
OVO-SVM	60.40	87.56	92.63	75.07	94.58	84.52	86.67	93.18
OVA-SVM	67.06	89.26	92.40	74.93	95.33	82.62	90	93.18



(a) (b)

Figure 5.12: Confusion matrix for six expressions on (a) KDEF dataset using MTP method (b) ADFES dataset using $WGFD_v$ method with prime weights

methods is shown. In tables 5.8 and 5.9, the proposed methods and their recognition accuracy has been highlighted in bold. The comparison analysis of proposed method with the existing variants of binary patterns on WSEFEP and ADFES datasets is shown in figure 5.13. From table 5.8, the proposed methods outperformed the existing variants of binary patterns on all other datasets except KDEF dataset. Although LDDSCP method achieved 0.24% more than the proposed PGBP method in case of KDEF dataset, the proposed methods are better as they achieved better results on other datasets also. From table 5.9, for different 'in the lab' datasets, the proposed methods outperformed the existing FER methods.

5.4.3 Experiments for Seven Expressions

The experiments for seven expressions have been conducted on different FER datasets using WGFD, PGBP and LTrP methods. The proposed WGFD methods have been implemented with different weights the results have been tabulated. In table 5.10, for each dataset, the recognition accuracy comparison of $WGFD_h$ with different weights using OVO-SVM classifier, in table 5.11, the recognition accuracy comparison of $WGFD_h$ with different weights using OVA-SVM classifier is shown. In table 5.12, the recognition accu-

Table 5.8: Comparison analysis with existing variants of binary patterns for six expressions for different 'in the lab' datasets

Method	JAFFE	MUG	CK+	OULU	TFEID	KDEF	WSEFEP	ADFES
LBP [46]	56.66	82.65	89.97	75.34	92.42	80.95	87.22	90.16
LDP [97]	52.77	82.87	90.84	72.43	93.67	80.95	86.67	90.91
LDN [31]	56.66	81.96	88.84	72.29	93.33	82.62	87.22	88.64
CSLBP [28]	53.28	83.94	90.68	71.53	94.25	81.67	84.44	87.12
LGC [84]	58.89	86.22	89.61	73.47	95.00	83.81	87.78	90.91
LDTP [32]	55.55	82.04	89.60	72.29	93.25	83.57	87.78	88.64
LDTeRP [33]	58.54	80.15	85.89	68.54	90.75	81.43	81.11	84.89
ALDP [35]	55.09	82.89	89.06	70.28	93.54	80.95	85.00	85.61
MSBP [88]	56.75	85.78	90.58	73.41	94.25	83.10	86.67	90.15
LDSP [34]	56.70	85.19	91.54	68.47	94.50	82.38	85.00	86.36
LDDSCP [99]	59.55	86.15	89.61	73.47	95.50	84.76	83.89	89.39
RADAP [17]	57.22	83.48	90.63	75.90	94.17	82.38	87.78	91.67
LBP + LNeP [134]	61.29	86.52	92.21	75.00	93	83.57	88.89	90.15
$WGFD_h$	**65.06**	**88.30**	**92.53**	**77.08**	**95.42**	**84.52**	**91.11**	**93.94**
$WGFD_v$	**67.58**	**87.48**	**92.21**	**78.12**	**96.07**	**84.52**	**91.67**	**94.70**
PGBP	**62.75**	**88.89**	**91.66**	**76.32**	**95.29**	**84.52**	**90**	**93.18**
mTP	**62.87**	**89.26**	**92.95**	**76.39**	**94.50**	**82.86**	**90**	**93.94**
MTP	**67.06**	**89.26**	**92.40**	**74.93**	**95.33**	**84.52**	**90**	**93.18**

Table 5.9: Comparison with existing methods for six expressions for different 'in the lab' datasets

Dataset	Method	Accuracy	Dataset	Method	Accuracy
JAFFE	IFRBC [86]	62.29	MUG	ResNet50 [17]	86.88
	ResNet50 [17]	59.44		DAGSVM [129]	82.28
	LOOP [11]	58.72		LOOP [11]	85.33
	WLGC-HD [57]	60.7		HiNet [108]	87.8
	$WGFD_h$	**65.06**		**$WGFD_h$**	**88.30**
	$WGFD_v$	**67.58**		**$WGFD_v$**	**87.48**
	PGBP	**62.75**		**PGBP**	**88.89**
	mTP	**62.87**		**mTP**	**89.26**
	MTP	**67.06**		**MTP**	**89.26**
CK+	ResNet50 [17]	89.32	OULU	ResNet50 [17]	73.1
	HiNet [108]	91.40		HiNet [108]	74.3
	WLGC-HD [57]	72.80		VGG16 [17]	73.4
	$WGFD_h$	**92.53**		**$WGFD_h$**	**77.08**
	$WGFD_v$	**92.21**		**$WGFD_v$**	**78.12**
	PGBP	**91.66**		**PGBP**	**76.32**
	mTP	**92.95**		**mTP**	**76.39**
	MTP	**92.40**		**MTP**	**74.93**
TFEID	DSNGE [85]	93.89	KDEF	IFRBC [86]	77.98
	DAMCNN [102]	93.65		ICVR [86]	76.31
	LIoP + HOG [123]	93.50		HOG [108]	82.17
	$WGFD_h$	**95.42**		**$WGFD_h$**	**84.52**
	$WGFD_v$	**96.07**		**$WGFD_v$**	**84.52**
	PGBP	**95.29**		**PGBP**	**84.52**
	mTP	**94.50**		**mTP**	**82.86**
	MTP	**95.33**		**MTP**	**84.52**
WSEFEP	LOOP [11]	87.78	ADFES	LOOP [11]	91.67
	$WGFD_h$	**91.11**		**$WGFD_h$**	**93.94**
	$WGFD_v$	**91.67**		**$WGFD_v$**	**94.70**
	PGBP	**90**		**PGBP**	**93.18**
	mTP	**90**		**mTP**	**93.94**
	MTP	**90**		**MTP**	**93.18**

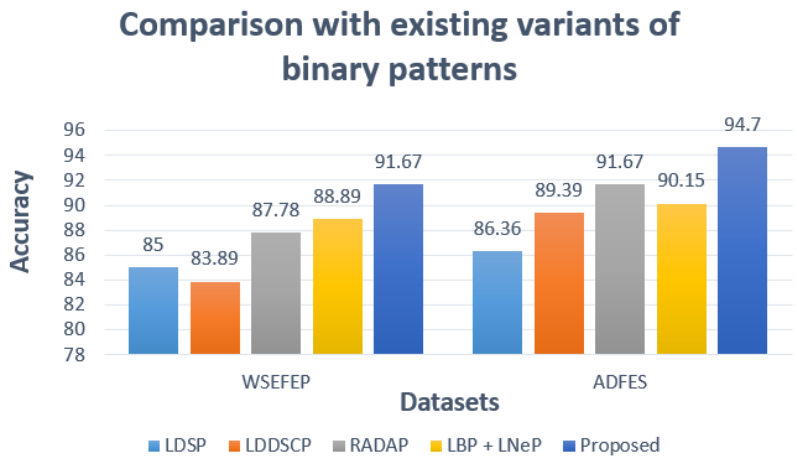


Figure 5.13: Comparison analysis of proposed method with existing variants of binary patterns for six expressions on WSEFEP and ADFES datasets

racy comparison of $WGFD_v$ with different weights using OVO-SVM classifier and in table 5.13, the recognition accuracy comparison of $WGFD_v$ with different weights using OVA-SVM classifier is shown. In tables 5.14, 5.15 and 5.16 the recognition accuracy of PGBP, mTP and MTP with OVO-SVM and OVA-SVM classifiers for different datasets is shown. Among the proposed methods with different weights, $WGFD_h$ method with prime weights achieved an optimal recognition accuracy of 88.10% on WSEFEP dataset. $WGFD_v$ method with fibonacci, natural and squares / fibonacci weights achieved an optimal recognition accuracy of 66.13%, 90.40% and 96.19% on JAFFE, CK+ and TFEID datasets respectively. PGBP method achieved an optimal recognition accuracy of 75.89% and 83.47% on OULU and KDEF datasets respectively. MTP and mTP methods achieved an optimal recognition accuracy of 87.11% and 94.16% on MUG and ADFES datasets respectively. PGBP and MTP methods achieved an optimal recognition accuracy of 77.74% and 99.83% on RAF and FERG datasets respectively.

The confusion matrix obtained using $WGFD_v$ method with fibonacci weights for JAFFE dataset is presented in figure 5.14(a) and for MUG dataset using MTP method is presented in figure 5.14(b). The confusion matrix obtained using $WGFD_h$ method with prime weights for WSEFEP dataset is presented in figure 5.15(a) and for FERG dataset using MTP method is presented in figure 5.15(b). In table 5.17, the comparison analysis of the proposed methods with the existing variants of binary patterns, implemented in our envi-

Table 5.10: Recognition accuracy of $WGFD_h$ with different weights for seven expressions using OVO-SVM classifier for different datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	58.57	60.50	58.53	60.85	60.93	59.09
MUG	81.84	82.67	81.71	80.32	81.90	81.14
CK+	86.61	86.46	86.67	86.25	86.77	86.70
OULU	73.27	73.69	73.69	73.04	73.57	72.50
TFEID	93.99	94.05	93.93	95.35	93.63	94.40
KDEF	81.22	80.82	80.61	79.59	80	80.41
WSEFEP	84.29	82.86	85.24	83.33	82.86	82.86
ADFES	91.56	92.21	92.86	92.21	92.21	89.61
RAF	74.64	74.09	73.60	71.32	74.77	72.69
FERG	99.57	94.46	99.29	98.20	99.20	98.31

Table 5.11: Recognition accuracy of $WGFD_h$ with different weights for seven expressions using OVA-SVM classifier for different datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	59.58	64.16	63.29	60.48	62.49	59.50
MUG	83.11	84.38	84.32	83.49	84.44	84.25
CK+	88.97	89.66	89.26	89.85	89.40	89.82
OULU	73.87	74.76	73.57	73.27	74.52	72.56
TFEID	95.83	96.13	94.50	94.08	95.42	95.33
KDEF	82.45	82.86	82.45	81.02	82.24	81.02
WSEFEP	87.62	85.71	88.10	86.67	86.67	85.71
ADFES	92.21	92.21	93.51	92.86	92.21	93.51
RAF	70.57	69.43	68.87	65.25	69.82	67.24
FERG	99.49	99.74	99.16	98.49	98.80	98.24

ronment setup is shown. In table 5.18, the comparison analysis of the proposed methods with the existing methods is shown. The comparison analysis for RAF and FERG datasets with the existing methods is reported in table 5.19. The comparison analysis of proposed method with the existing methods on RAF and FERG datasets is shown in figure 5.16. In tables 5.17, 5.18 and 5.19, the proposed methods and their recognition accuracy has been highlighted in bold. From table 5.17, the proposed methods outperformed the existing variants of binary patterns for different FER datasets. From table 5.18, for different 'in the lab' datasets, the proposed methods outperformed the existing FER methods. From table 5.19, the proposed PGBP and MTP methods outperformed the existing methods on RAF and FERG datasets respectively.

Table 5.12: Recognition accuracy of $WGFD_v$ with different weights for seven expressions using OVO-SVM classifier for different datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	61.86	62.34	60.91	60.87	61	61.40
MUG	81.71	81.90	82.60	79.81	82.22	80.83
CK+	86.95	86.64	87.21	87.37	87	86.74
OULU	72.98	73.57	73.69	73.33	73.96	72.69
TFEID	93.99	94.40	94.35	93.63	93.99	94.05
KDEF	81.84	81.22	80.61	81.22	81.22	81.02
WSEFEP	84.29	82.86	85.24	83.33	82.86	82.86
ADFES	90.91	90.91	90.26	89.61	90.26	88.96
RAF	75.68	74.90	74.25	72.56	74.98	74.45
FERG	98.14	98.14	98.14	98.14	98.14	98.14

Table 5.13: Recognition accuracy of $WGFD_v$ with different weights for seven expressions using OVA-SVM classifier for different datasets

Dataset	Binary	Fibonacci	Prime	Natural	Squares	Odd
JAFFE	62.36	66.13	65.67	61.45	64.29	62.42
MUG	84.69	83.68	84.13	82.60	82.79	82.24
CK+	88.97	89.93	89.37	90.40	89.41	89.78
OULU	74.29	74.29	74.46	74.40	73.93	72.50
TFEID	95.33	96.19	94.56	94.92	96.19	95.33
KDEF	83.47	82.86	82.24	82.45	83.27	82.05
WSEFEP	87.62	87.14	87.62	86.19	87.62	86.19
ADFES	93.51	91.56	92.86	92.86	92.21	92.86
RAF	71.25	70.08	69.39	66.98	71.74	69
FERG	98.74	99.13	98.14	98.14	98.14	98.14

Table 5.14: Recognition accuracy of PGBP for seven expressions for different datasets

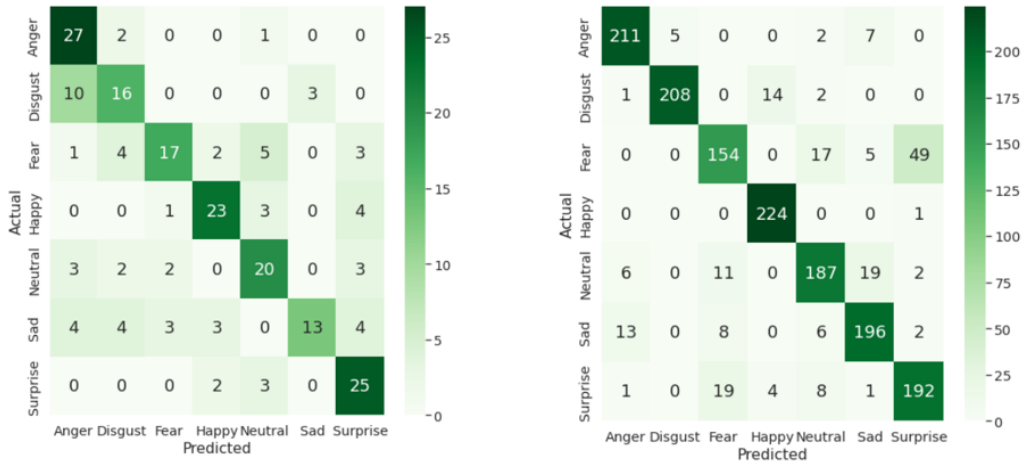
Classifier	JAFFE	MUG	CK+	OULU	TFEID	KDEF	WSEFEP	ADFES	RAF	FERG
OVO-SVM	58.59	82.92	86.87	73.85	93.57	81.43	84.29	92.21	77.74	99.47
OVA-SVM	62.78	86.79	90.09	75.89	95	83.47	86.67	93.51	75	98.14

Table 5.15: Recognition accuracy of mTP for seven expressions for different datasets

Classifier	JAFFE	MUG	CK+	OULU	TFEID	KDEF	WSEFEP	ADFES	RAF	FERG
OVO-SVM	59.09	81.97	86.56	73.51	94.35	81.22	84.29	90.91	75	98.29
OVA-SVM	60.54	85.02	88.40	74.46	95	81.63	85.71	94.16	70.05	98.58

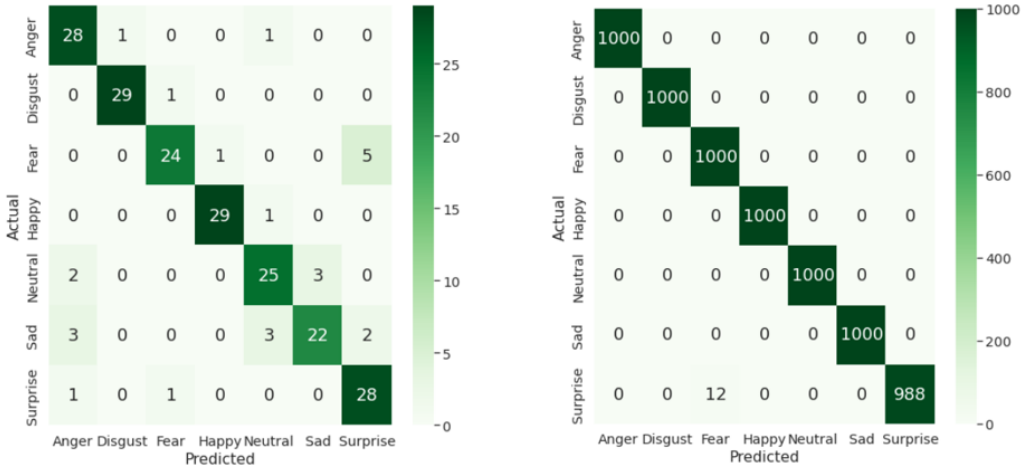
Table 5.16: Recognition accuracy of MTP for seven expressions for different datasets

Classifier	JAFFE	MUG	CK+	OULU	TFEID	KDEF	WSEFEP	ADFES	RAF	FERG
OVO-SVM	61.30	83.37	86.51	72.41	93.57	82.04	82.86	92.86	76.17	99.83
OVA-SVM	64.14	87.11	89.91	74.40	95.36	81.22	85.71	92.86	71.32	98.14



(a) (b)

Figure 5.14: Confusion matrix for seven expressions on (a) JAFFE dataset using $WGFD_v$ method with fibonacci weights (b) MUG dataset using MTP method



(a) (b)

Figure 5.15: Confusion matrix for seven expressions on (a) WSEFEP dataset using $WGFD_h$ method with prime weights (b) FERG dataset using MTP method

Table 5.17: Comparison analysis with existing variants of binary patterns for seven expressions for different datasets

Method	JAFFE	MUG	CK+	OULU	TFEID	KDEF	WSEFEP	ADFES
LBP [46]	53.65	76.16	83.96	65.71	92.02	78.16	82.38	87.66
LDP [97]	52.10	78.70	84.80	69.16	86.20	80.61	80.95	90.26
LDN [31]	54.87	77.85	83.35	70.89	93.51	80.75	83.81	91.56
CSLBP [28]	52.40	79.37	85.51	57.98	89.44	80.20	79.44	86.36
LGC [84]	57.59	82.03	86.71	71.13	93.43	81.02	84.29	90.91
LDTP [32]	51.32	78.70	83.08	68.86	93.15	80.81	85.71	85.71
LDTeRP [33]	51.70	78.11	81.40	64.53	90.18	77.35	79.52	79.87
ALDP [35]	51.53	77.59	85.61	67.74	92.36	76.53	80.48	83.77
MSBP [88]	56.81	81.40	86.04	73.75	93.27	81.22	83.73	91.56
LDSP [34]	52.49	80.63	84.19	63.81	93.15	80.61	85.00	85.06
LDDSCP [99]	57.37	80.95	84.93	69.57	90.95	81.84	82.38	88.96
RADAP [17]	56.20	80.26	84.60	74.34	93.27	80.20	87.42	90.91
LBP + LNeP [134]	59.04	81.45	86.30	73.20	93.27	81.84	86.19	90.26
$WGFD_h$	**64.16**	**84.44**	**89.82**	**74.76**	**96.13**	**82.86**	**88.10**	**93.51**
$WGFD_v$	**66.13**	**84.69**	**90.40**	**74.46**	**96.17**	**83.47**	**87.62**	**93.51**
PGBP	**62.78**	**86.79**	**90.09**	**75.89**	**95**	**83.47**	**86.67**	**93.51**
mTP	**60.54**	**85.62**	**88.40**	**74.46**	**95**	**81.63**	**85.71**	**94.16**
MTP	**64.14**	**87.11**	**89.91**	**74.40**	**95.36**	**82.04**	**85.71**	**92.86**

Table 5.18: Comparison with existing methods for seven expressions

Dataset	Method	Accuracy	Dataset	Method	Accuracy
JAFFE	PCANet [117]	58.35	MUG	CBA [90]	78.57
	ResNet50 [17]	57.13		ResNet50 [17]	85.58
	LOOP [11]	59.67		LOOP [11]	79.68
	WLGC-HD [57]	58.2		HiNet [108]	87.2
	$WGFD_h$	**64.16**		**$WGFD_h$**	**84.44**
	$WGFD_v$	**66.13**		**$WGFD_v$**	**84.69**
	PGBP	**62.78**		**PGBP**	**86.79**
	mTP	**60.54**		**mTP**	**85.62**
	MTP	**64.14**		**MTP**	**87.11**
CK+	ResNet50 [17]	87.31	OULU	ResNet50 [17]	65.4
	HiNet [108]	88.6		HiNet [108]	72
	DLFS [107]	83.72		VGG19 [17]	70.5
	$WGFD_h$	**89.82**		**$WGFD_h$**	**74.46**
	$WGFD_v$	**90.40**		**$WGFD_v$**	**74.46**
	PGBP	**90.09**		**PGBP**	**75.89**
	mTP	**88.40**		**mTP**	**74.46**
	MTP	**89.91**		**MTP**	**74.40**
TFEID	DAMCNN [102]	93.36	KDEF	DLFS [107]	78.60
	Pyramid+SBDT [87]	93.38		PCANet [117]	69.59
	MSDV [132]	93.50		SAFL [108]	81.22
	$WGFD_h$	**96.13**		**$WGFD_h$**	**82.86**
	$WGFD_v$	**96.17**		**$WGFD_v$**	**83.47**
	PGBP	**95**		**PGBP**	**83.47**
	mTP	**95**		**mTP**	**81.63**
	MTP	**95.36**		**MTP**	**82.04**
WSEFEP	LOOP[11]	84.68	ADFES	AFM [103]	92.70
	$WGFD_h$	**88.10**		**$WGFD_h$**	**93.51**
	$WGFD_v$	**87.62**		**$WGFD_v$**	**93.51**
	PGBP	**86.67**		**PGBP**	**93.51**
	mTP	**85.71**		**mTP**	**94.16**
	MTP	**85.71**		**MTP**	**92.86**

Table 5.19: Comparison with existing methods for seven expressions on RAF and FERG datasets

Dataset	Method	Accuracy	Dataset	Method	Accuracy
RAF	DLPCNN [55]	74.20	FERG	Deep Expr [56]	89.02
	ICID Fusion [101]	75.40		Ensemble Multi-Feature [17]	97
	Sadeghi et al. [93]	76.23		Adversarial NN [105]	98.2
	DCNN+RLPS [108]	72.84		Deep Emotion [106]	99.3
	IFSL [120]	76.9		LBP-AW [16]	96.7
	$WGFD_h$	**74.77**		**$WGFD_h$**	**99.74**
	$WGFD_v$	**75.68**		**$WGFD_v$**	**99.13**
	PGBP	**77.74**		**PGBP**	**99.47**
	mTP	**75**		**mTP**	**98.58**
	MTP	**76.17**		**MTP**	**99.83**

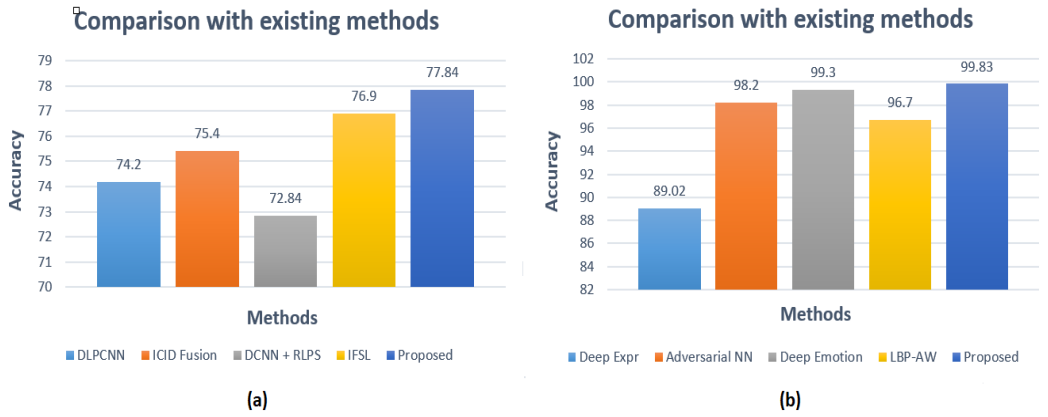


Figure 5.16: Comparison analysis of proposed method with existing methods for seven expressions on (a) RAF and (b) FERG datasets

Table 5.20: Recognition accuracy comparison using WGFD methods for eight expressions with different weights on TFEID dataset

Classifier	Method	Binary	Fibonacci	Prime	Natural	Squares	Odd
OVO-SVM	$WGFD_h$	92.32	91.73	92	90.61	92	91.69
	$WGFD_v$	91.03	90.44	91.27	91.23	91.31	91.31
OVA-SVM	$WGFD_h$	93.42	92.55	92.51	93.18	92.83	93.73
	$WGFD_v$	93.73	93.45	92.58	92.86	93.42	93.42

5.4.4 Experiments for Eight Expressions

The proposed WGFD methods have been implemented with different weights and the results have been tabulated in table 5.20. In table 5.21, the recognition accuracy comparison using PGBP, mTP and MTP methods is shown. Among the proposed methods, MTP method achieved an optimal recognition accuracy of 95.66%. The comparison analysis of the proposed methods with the existing variants of binary patterns is reported in the second column of table 5.23. In table 5.23, the proposed methods and their recognition accuracy has been highlighted in bold. From table 5.23, the experimental results indicate that the proposed methods outperformed the existing variants of binary patterns in terms of recognition accuracy.

Table 5.21: Recognition accuracy comparison using PGBP, mTP and MTP methods for eight and ten expressions

Classifier	Method	Eight Expressions	Ten Expressions
OVO-SVM	PGBP	92.21	87.37
	mTP	91.66	85.15
	MTP	93.77	87.30
OVA-SVM	PGBP	95.28	84.75
	mTP	94.01	83.33
	MTP	95.56	83.79

Table 5.22: Recognition accuracy comparison using WGFD methods for ten expressions with different weights on ADFES dataset

Classifier	Method	Binary	Fibonacci	Prime	Natural	Squares	Odd
OVO-SVM	$WGFD_h$	85.71	85.71	86.16	84.70	87.98	84.70
	$WGFD_v$	86.16	85.25	84.75	85.15	85.66	84.29
OVA-SVM	$WGFD_h$	83.28	83.74	84.14	85.20	84.65	82.83
	$WGFD_v$	83.74	83.74	83.69	83.29	83.28	82.88

5.4.5 Experiments for Ten Expressions

The proposed WGFD methods have been implemented with different weights and the results have been tabulated in table 5.22. In table 5.21, the recognition accuracy comparison using PGBP, mTP and MTP methods is shown. Among the proposed methods, $WGFD_h$ method with squares weights achieved an optimal recognition accuracy of 87.98%. The comparison analysis of the proposed methods with the existing variants of binary patterns is reported in the third column of table 5.23. From table 5.23, the experimental results indicate that the proposed methods outperformed the existing variants of binary patterns in terms of recognition accuracy .

5.5 Summary

An appropriate representation of facial features can significantly aid in improving the performance of an FER system. In this regard, new texture based feature descriptors inspired by the shape of various graphs such as Windmill graph, GPG and Triangle graph have been proposed in this Chapter. Each of $WGFD_h$ and $WGFD_v$ methods generate two feature

Table 5.23: Comparison analysis with existing variants of binary patterns for eight and ten expressions on TFEID and ADFES datasets

Method	Eight Expressions	Ten Expressions
LBP [46]	90.95	83.54
LDP [97]	91.17	85.25
LDN [31]	88.06	85.20
CSLBP [28]	89.45	80.61
LGC [84]	89.95	85.20
LDTP [32]	86.29	82.68
LDTerP [33]	88.59	76.53
ALDP [35]	90.36	78.40
MSBP [88]	89.54	87.17
LDSP [34]	86.68	79.91
LDDSCP [99]	87.66	84.96
RADAP [17]	90.56	84.75
LBP + LNeP [134]	90.81	87.07
$WGFD_h$	**93.73**	**87.98**
$WGFD_v$	**93.73**	**86.16**
PGBP	**95.28**	**87.37**
mTP	**94.01**	**85.15**
MTP	**95.56**	**87.30**

codes by encoding both the adjacent pixel relationship and the neighboring pixel relationship in a local neighborhood. The concept of using different weights (binary, fibonacci, prime, natural, squares and odd) have been applied to the proposed WGFD methods for determining the optimal recognition accuracy. PGBP extracts three feature codes based on the vertices and edges of Generalised Petersen graph (GPG(6,2)) in a local neighborhood. LTrP (mTP and MTP) methods generate two feature codes by considering the triangles in both vertical and horizontal directions. The features are extracted both in clockwise and counter clockwise directions using the proposed PGBP and LTrP methods. In this Chapter, the experiments have been conducted using different weights for WGFD methods and using binary weights for PGBP and LTrP methods. From the experimental results, an observation has been made that proposed methods outperformed the standard existing methods proving the robustness and efficiency of the proposed descriptors.

Chapter 6

Feed Forward Neural Network Structure Inspired Hand-crafted Feature Descriptors

Automatic FER has become essential today as it has many applications in real time such as animation, driver mood detection, lie detection, clinical psychology etc [57]. The effectiveness of FER systems mainly depends on the extracted features. For extracting distinctive features with low dimensions, new local texture based image descriptors named FFNND have been proposed in this Chapter for recognizing facial expressions. The main contributions in this Chapter are summarized as follows:

- Novel texture based feature extraction methods named FFNND ($FFFND_1$, $FFNNF_2$), inspired by the structure of a feed forward neural network have been proposed for extracting the facial features in a local neighborhood. The weights in the network are calculated based on the neighboring pixel values.
- In order to find the optimal recognition accuracy of the proposed FFNND methods, the experiments have been performed with logsig and tanh activation functions under varying block sizes using multi-class SVM.

In section 6.1, the feature descriptors named FFNND inspired from the structure of a feed forward neural network have been proposed. In section 6.2, the experimental results

corresponding to $FFNND_1$ and $FFNND_2$ have been presented and analyzed. In section 6.3, the contributions in this Chapter have been summarized.

6.1 Feed Forward Neural Network Inspired Feature Descriptors (FFNND)

Initially, Tuncer et al. [173] proposed Neural Network based Image Descriptor (NND) for texture recognition. The proposed methods have been inspired by the structure of a feed forward neural network, as like NND and the main objective of the proposed methods is to extract salient features in a local neighborhood. For extracting discriminative features, two feature descriptors named $FFNND_1$ and $FFNND_2$ have been proposed in this Chapter. $FFNND_1$ extracts three feature codes by capturing the adjacent pixel relationship based on multi-distance information as like LMeP, whereas, $FFNND_2$ extracts two feature codes by capturing the relationship between the pixels in a circular neighborhood located at a radius (rd = 2), as like MTP.

6.1.1 Feature extraction using $FFNND_1$

Feature extraction using $FFNND_1$ involves dividing an image into 3 x 3 overlapping blocks and the pixels within each block are considered to create feed forward neural networks. LMeP method [168] generates three feature codes based on the adjacent pixel relationship, corresponding to multi-distance information ($d \in [1,3]$). Hence, for each distance, one neural network is created and thus, finally three neural networks are created. These networks use the pixels of the 3 x 3 overlapping blocks and the weights of these networks are computed based on the neighboring pixels in the network and the signum function (s(x)). In these networks, logistic sigmoid (logsig) and tangent hyperbolic (tanh) functions are utilized as activation function. Finally, the histogram features obtained from all the three networks are concatenated to obtain a final feature vector. The complete process of feature extraction using $FFNND_1$ is explained in the following steps:

Step 1: Divide the input image into 3 x 3 overlapping blocks. These 3 x 3 blocks are

chosen in order to capture finer appearance changes with respect to facial expressions in a local neighborhood [11, 15, 17, 33].

Step 2: The pixels in a 3 x 3 neighborhood are normalized based on the following formula, as shown in eq.(6.1):

$$p_{\mathrm{m}} = \frac{p_{\mathrm{m}}}{maximum(p_{\mathrm{m}})}, \quad where\ p_{\mathrm{m}} = \{r_1, b_1, r_2, b_2, r_3, b_3, r_4, b_4, p_{\mathrm{c}}\} \tag{6.1}$$

Step 3: LMeP compares adjacent pixels for feature extraction, whereas, NND considers the structure of a feed forward neural network for feature extraction. The proposed $\mathrm{FFNND_1}$ feature descriptor is developed by combining the properties of both of these descriptors.

$$mesh_{\mathrm{a_1}} = \sigma(r_1 * w_{\mathrm{r_1b_2}} + b_1 * w_{\mathrm{b_1b_2}} + r_2 * w_{\mathrm{r_2b_2}} + p_{\mathrm{c}}) \tag{6.2}$$

$$mesh_{\mathrm{b_1}} = \sigma(r_1 * w_{\mathrm{r_1r_3}} + b_1 * w_{\mathrm{b_1r_3}} + r_2 * w_{\mathrm{r_2r_3}} + p_{\mathrm{c}}) \tag{6.3}$$

$$mesh_{\mathrm{c_1}} = \sigma(mesh_{\mathrm{a_1}} * w_{\mathrm{b_2b_3}} + mesh_{\mathrm{b_1}} * w_{\mathrm{r_3b_3}} + p_{\mathrm{c}}) \tag{6.4}$$

$$mesh_{\mathrm{d_1}} = \sigma(mesh_{\mathrm{a_1}} * w_{\mathrm{b_2r_4}} + mesh_{\mathrm{b_1}} * w_{\mathrm{r_3r_4}} + p_{\mathrm{c}}) \tag{6.5}$$

$$mesh_{\mathrm{e_1}} = \sigma(mesh_{\mathrm{a_1}} * w_{\mathrm{b_2b_4}} + mesh_{\mathrm{b_1}} * w_{\mathrm{r_3b_4}} + p_{\mathrm{c}}) \tag{6.6}$$

$$fv_1 = \sigma(mesh_{\mathrm{c_1}} * w_{\mathrm{b_3p_c}} + mesh_{\mathrm{d_1}} * w_{\mathrm{r_4p_c}} + mesh_{\mathrm{e_1}} * w_{\mathrm{b_4p_c}} + p_{\mathrm{c}}) \tag{6.7}$$

$$w_{\mathrm{u_yv_z}} = \begin{cases} 1, & \text{if } u_{\mathrm{y}} \geq v_{\mathrm{z}} \\ 0, & otherwise \end{cases} \tag{6.8}$$

The process of feature extraction using $\mathrm{FFNND_1}$, corresponding to distance (d = 1) is shown in eqs.(6.2) to (6.8). Here, $\mathrm{r_1}$, $\mathrm{b_1}$, $\mathrm{r_2}$, $\mathrm{b_2}$, $\mathrm{r_3}$, $\mathrm{b_3}$, $\mathrm{r_4}$, $\mathrm{b_4}$ correspond to the neighboring pixels surrounding the center pixel ($\mathrm{p_c}$), set $\mathrm{w_{u_yv_z}}$ represents the weights. Mesh ($\mathrm{mesh_{a_1}}$, $\mathrm{mesh_{b_1}}$, $\mathrm{mesh_{c_1}}$, $\mathrm{mesh_{d_1}}$, $\mathrm{mesh_{e_1}}$) corresponds to the hidden layer outputs and $\mathrm{fv_1}$ corresponds to the feature code for the network formed by the pixels of $\mathrm{LMeP_1}$. The $\sigma(.)$ function corresponds to the activation function. At each step, $\mathrm{p_c}$ is considered as the bias.

$$mesh_{\mathrm{a_2}} = \sigma(r_1 * w_{\mathrm{r_1r_4}} + r_2 * w_{\mathrm{r_2r_4}} + r_3 * w_{\mathrm{r_3r_4}} + p_{\mathrm{c}}) \tag{6.9}$$

$$mesh_{\mathrm{b_2}} = \sigma(r_1 * w_{\mathrm{r_1b_1}} + r_2 * w_{\mathrm{r_2b_1}} + r_3 * w_{\mathrm{r_3b_1}} + p_{\mathrm{c}}) \tag{6.10}$$

$$mesh_{\mathrm{c_2}} = \sigma(mesh_{\mathrm{a_2}} * w_{\mathrm{r_4b_2}} + mesh_{\mathrm{b_2}} * w_{\mathrm{b_1b_2}} + p_{\mathrm{c}}) \tag{6.11}$$

$$mesh_{\mathrm{d_2}} = \sigma(mesh_{\mathrm{a_2}} * w_{\mathrm{r_4b_3}} + mesh_{\mathrm{b_2}} * w_{\mathrm{b_1b_3}} + p_{\mathrm{c}}) \tag{6.12}$$

$$mesh_{\mathrm{e_2}} = \sigma(mesh_{\mathrm{a_2}} * w_{\mathrm{w_{r_4b_4}}} + mesh_{\mathrm{b_2}} * w_{\mathrm{b_1b_4}} + p_{\mathrm{c}}) \tag{6.13}$$

$$fv_2 = \sigma(mesh_{\mathrm{c_2}} * w_{\mathrm{b_2p_c}} + mesh_{\mathrm{d_2}} w_{\mathrm{b_3p_c}} + mesh_{\mathrm{e_2}} * w_{\mathrm{b_4p_c}} + p_{\mathrm{c}}) \tag{6.14}$$

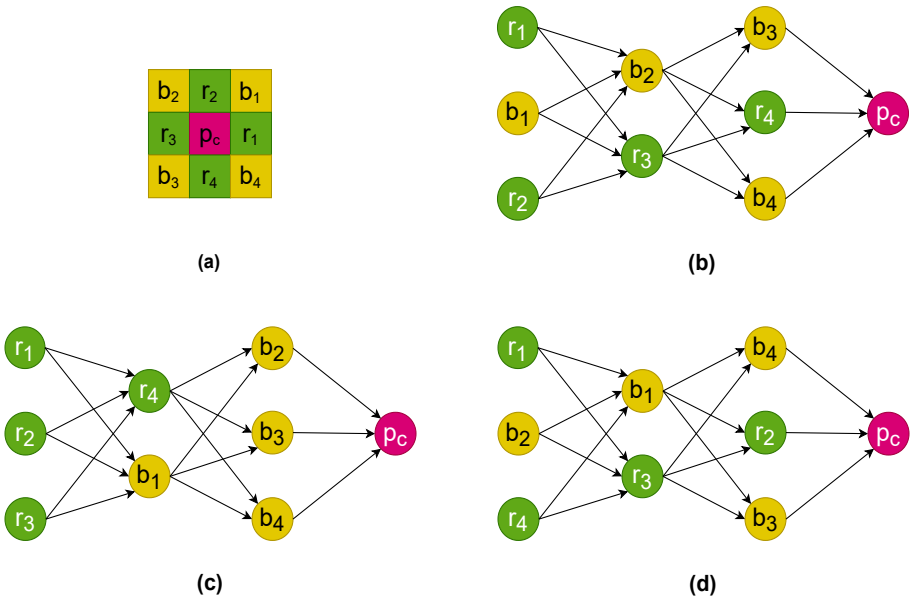


Figure 6.1: The proposed feature descriptor network (FFNND$_1$) (a) 3 x 3 sample block (b) Network constructed by considering adjacent pixels corresponding to d = 1 (c) Network constructed by considering adjacent pixels corresponding to d = 2 (d) Network constructed by considering adjacent pixels corresponding to d = 3.

The process of feature extraction using FFNND$_1$, corresponding to distance (d = 2) is shown in eqs.(6.9) to (6.14). Mesh (mesh$_{a_2}$, mesh$_{b_2}$, mesh$_{c_2}$, mesh$_{d_2}$, mesh$_{e_2}$) corresponds to the hidden layer outputs and fv$_2$ corresponds to the feature code for the network formed by the pixels of LMeP$_2$.

$$mesh_{a_3} = \sigma(r_1 * w_{r_1b_1} + b_2 * w_{b_2b_1} + r_4 * w_{r_4b_1} + p_c) \tag{6.15}$$

$$mesh_{b_3} = \sigma(r_1 * w_{r_1r_3} + b_2 * w_{b_2r_3} + r_4 * w_{r_4r_3} + p_c) \tag{6.16}$$

$$mesh_{c_3} = \sigma(mesh_{a_3} * w_{b_1b_4} + mesh_{b_3} * w_{r_3b_4} + p_c) \tag{6.17}$$

$$mesh_{d_3} = \sigma(mesh_{a_3} * w_{b_1r_2} + mesh_{b_3} * w_{r_3r_2} + p_c) \tag{6.18}$$

$$mesh_{e_3} = \sigma(mesh_{a_3} * w_{b_1r_3} + mesh_{b_3} * w_{r_3b_3} + p_c) \tag{6.19}$$

$$fv_3 = \sigma(mesh_{c_3} * w_{b_4p_c} + mesh_{d_3} * w_{r_2p_c} + mesh_{e_3} * w_{b_3p_c} + p_c) \tag{6.20}$$

The process of feature extraction using FFNND$_1$, corresponding to distance (d = 3) is shown in eqs.(6.15) to (6.20). Mesh (mesh$_{a_3}$, mesh$_{b_3}$, mesh$_{c_3}$, mesh$_{d_3}$, mesh$_{e_3}$) corresponds to the hidden layer outputs and fv$_3$ corresponds to the feature code for the network formed by the pixels of LMeP$_3$. In this Chapter, as a part of experimental evaluation, different acti-

vation functions such as logistic sigmoid (logsig) and tangent hyperbolic (tanh) have been considered. The mathematical equations corresponding to the logsig and tanh activation functions are shown in eqs.(6.21) and (6.22).

$$logsig(g) = \frac{1}{(1 + e^{-g})} \tag{6.21}$$

$$tanh(g) = \frac{(e^{g} - e^{-g})}{(e^{g} + e^{-g})} \tag{6.22}$$

where, logsig(.) and tanh(.) represents logistic sigmoid and tangent hyperbolic activation functions respectively.

Step 4: The feature codes fv_1, fv_2 and fv_3 are normalized based on the following formulae, as shown in eqs.(6.23) to (6.25).

$$fv_1 = round(fv_1 * scale) \tag{6.23}$$

$$fv_2 = round(fv_2 * scale) \tag{6.24}$$

$$fv_3 = round(fv_3 * scale) \tag{6.25}$$

where round(.) is the rounding function used for integer conversion and scale denotes the normalization range. As the values are normalized in step 2, fv_1, fv_2 and fv_3 are in the range [0,1]. As the values are in the range [0,1], they have been rounded and multiplied by the scale. The scale is chosen empirically as 10 in order to greatly reduce the feature vector length.

Step 5: Repeat the steps 2-4 for all the 3 x 3 blocks and generate three feature response maps for an image.

Step 6: Extract the block wise histograms from the each of the obtained feature response maps, as shown in eqs.(6.26) to (6.28). Here, $hist_{m_1}$, $hist_{m_2}$ and $hist_{m_3}$ correspond to the block wise histograms obtained from fv_1, fv_2 and fv_3 respectively.

$$hist_{m_1} = Hist(fv_1) \tag{6.26}$$

$$hist_{m_2} = Hist(fv_2) \tag{6.27}$$

$$hist_{m_3} = Hist(fv_3) \tag{6.28}$$

Step 7: Concatenate all the histograms to form final feature vector (fv_{ffnnd_1}), as shown in eq.(6.29).

$$fv_{ffnnd_1} = hist_{m_1} \cup hist_{m_2} \cup hist_{m_3} \tag{6.29}$$

where, fv_{ffnnd_1} is the final fv obtained using $FFNND_1$ method. Feature extraction using the proposed $FFNND_1$ method is shown graphically in figure 6.1. As shown in figure 6.1, the proposed $FFNND_1$ method generates three feature codes, as it utilizes three networks. The process of feature extraction using $FFNND_1$ method is shown in algorithm 6.1. The value of 'L' in algorithm 6.1 is 10.

Algorithm 6.1 Feature Extraction through $FFNND_1$

Input: An Input image (Img) of size N x N
Output: Feature vector (fv_{ffnnd_1}) of size $\lceil$(N-2)/C$\rceil$ * $\lceil$(N-2)/C$\rceil$ * 3 * L

procedure $FFNND_1$(Img)
  Initialization: $fv_1, fv_2, fv_3 \leftarrow \{\ \}$
  Load an input image (Img)
  for all a $\in$ range(1,N-2) **do**
    for all b $\in$ range(1,N-2) **do**
      Block = img(a:a+2,b:b+2)
      Assign the pixel values to Rook and Bishop in the 3 x 3 block.
      Calculate fv_1 using eqs.(6.2) to (6.8)
      Calculate fv_2 using eqs.(6.9) to (6.14)
      Calculate fv_1 using eqs.(6.15) to (6.20)
      Normalize fv_1, fv_2 and fv_3 as shown in eqs. (6.23) to (6.25)
    end for
  end for
  Create three feature response maps obtained from fv_1, fv_2 and fv_3 by reshaping them to (N-2) x (N-2).
  Each feature response map is partitioned into C x C non-overlapping blocks.
  Histograms are extracted block wise for all the three feature response maps using eqs.(6.26) to (6.28).
  Concatenate all the histograms to obtain fv_{ffnnd_1} using eq.(6.29).
  return fv_{ffnnd_1}
end procedure

6.1.2 Feature extraction using $FFNND_2$

Feature extraction using $FFNND_2$ involves dividing an image into 5 x 5 overlapping blocks and the pixels within each block are considered to create feed forward neural networks. $FFNND_2$ method considers circular neighborhood of radius (rd = 2), as like MTP. For feature extraction through $FFNND_2$, only twelve pixels surrounding the pixel p_c are considered. These twelve pixels are logically divided into two groups namely horizontal group

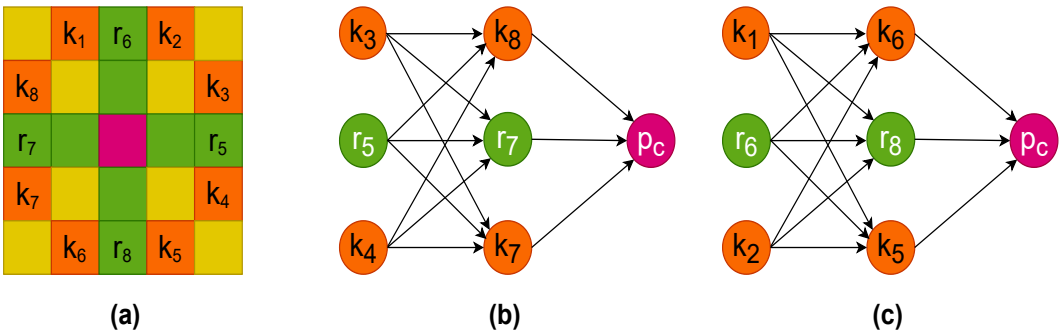


Figure 6.2: The proposed feature descriptor network ($FFNND_2$) (a) 5 x 5 sample block corresponding to rd = 2 (b) Network constructed by considering pixels in horizontal direction (c) Network constructed by considering pixels in vertical direction.

(k_3, r_5, k_4, k_8, r_7, k_7) and vertical group (k_1, r_6, k_2, k_6, r_8, k_5). Two neural networks are created by separately considering the horizontal group and vertical group of pixels. A sample 5 x 5 block corresponding to circular neighborhood (rd = 2) is shown in figure 6.2(a). In figure 6.2(b), the network created using horizontal groups of pixels and the pixel p_c is shown. In figure 6.2(c), the network created using vertical groups of pixels and the pixel p_c is shown. These networks use the pixels of the 5 x 5 circular neighborhood and the weights of these networks are computed based on the neighboring pixels in the network and the signum function (s(x)). In these networks, logistic sigmoid (logsig) and tangent hyperbolic (tanh) functions are utilized as activation function. Finally, the histogram features obtained from all the two networks are concatenated to obtain a final feature vector. The complete process of feature extraction using $FFNND_2$ is explained in the following steps:

Step 1: Divide the input image into 5 x 5 overlapping blocks. These 5 x 5 blocks are chosen for feature extraction as like MTP.

Step 2: The pixels in a 5 x 5 neighborhood are normalized based on the following formula, as shown in eq.(6.30):

$$p_{\mathrm{m}} = \frac{p_{\mathrm{m}}}{maximum(p_{\mathrm{m}})}, \quad where\ p_{\mathrm{m}} = \{r_5, k_3, k_2, r_6, k_1, k_8, r_7, k_7, k_6, r_8, k_5, k_4\} \qquad (6.30)$$

Step 3: MTP compares adjacent pixels for feature extraction by drawing inspiration from the shape of a triangle, whereas, NND considers the structure of a feed forward neural network for feature extraction. The proposed $FFNND_2$ method is developed by considering

the pixels in a circular neighborhood as like MTP and constructs networks as like NND.

$$mesh_{a_4} = \sigma(k_3 * w_{k_3k_8} + r_5 * w_{r_5k_8} + k_4 * w_{k_4k_8} + p_c) \tag{6.31}$$

$$mesh_{b_4} = \sigma(k_3 * w_{k_3r_7} + r_5 * w_{r_5r_7} + k_4 * w_{k_4r_7} + p_c) \tag{6.32}$$

$$mesh_{c_4} = \sigma(k_3 * w_{k_3k_7} + r_5 * w_{r_5k_7} + k_4 * w_{k_4k_7} + p_c) \tag{6.33}$$

$$fv_4 = \sigma(mesh_{a_4} * w_{k_8p_c} + mesh_{b_4} * w_{r_7p_c} + mesh_{c_4} * w_{k_7p_c} + p_c) \tag{6.34}$$

The process of feature extraction using $FFNND_2$, corresponding to horizontal group of pixels is shown in eqs.(6.31) to (6.34). Mesh ($mesh_{a_4}$, $mesh_{b_4}$ and $mesh_{c_4}$ corresponds to the hidden layer outputs and fv_4 corresponds to the feature code for the network formed by the considering horizontal group of pixels. At each step, p_c is considered as the bias.

$$mesh_{a_5} = \sigma(k_1 * w_{k_1k_6} + r_6 * w_{r_6k_6} + k_2 * w_{k_2k_6} + p_c) \tag{6.35}$$

$$mesh_{b_5} = \sigma(k_1 * w_{k_1r_8} + r_6 * w_{r_6r_8} + k_2 * w_{k_2r_8} + p_c) \tag{6.36}$$

$$mesh_{c_5} = \sigma(k_1 * w_{k_1k_5} + r_6 * w_{r_6k_5} + k_2 * w_{k_2k_5} + p_c) \tag{6.37}$$

$$fv_5 = \sigma(mesh_{a_5} * w_{k_6p_c} + mesh_{b_5} * w_{r_8p_c} + mesh_{c_5} * w_{k_5p_c} + p_c) \tag{6.38}$$

The process of feature extraction using $FFNND_2$, corresponding to vertical group of pixels is shown in eqs.(6.35) to (6.38). Mesh ($mesh_{a_5}$, $mesh_{b_5}$ and $mesh_{c_5}$ corresponds to the hidden layer outputs and fv_5 corresponds to the feature code for the network formed by the considering vertical group of pixels. At each step, p_c is considered as the bias. In this Chapter, as a part of experimental evaluation, different activation functions such as logsig and tanh have been considered. The mathematical equations corresponding to the logsig and tanh activation functions are shown in eqs.(6.21) and (6.22).

Step 4: The feature codes fv_4 and fv_5 are normalized based on the following formulae, as shown in eq.(6.39) and eq.(6.40).

$$fv_4 = round(fv_4 * scale) \tag{6.39}$$

$$fv_5 = round(fv_4 * scale) \tag{6.40}$$

where round(.) is the rounding function used for integer conversion and scale denotes the normalization range. As the values are normalized in step 2, fv_4 and fv_5 are in the range [0,1]. As the values are in the range [0,1], they have been rounded and multiplied by the scale. The scale is chosen empirically as 10 in order to greatly reduce the feature vector

length.

Step 5: Repeat the steps 2-4 for all the 5 x 5 blocks and generate two feature response maps for an image.

Step 6: Extract the block wise histograms from both the obtained feature response maps, as shown in eqs.(6.41) and (6.42). Here, hist_{m_4} and hist_{m_5} correspond to the block wise histograms obtained from fv_4 and fv_5 respectively.

$$hist_{\text{m}_4} = Hist(fv_4) \tag{6.41}$$

$$hist_{\text{m}_5} = Hist(fv_5) \tag{6.42}$$

Step 7: Concatenate all the histograms to form final feature vector ($\text{fv}_{\text{ffnnd}_2}$), as shown in eq.(6.43).

$$fv_{\text{ffnnd}_2} = hist_{\text{m}_4} \cup hist_{\text{m}_5} \tag{6.43}$$

where, $\text{fv}_{\text{ffnnd}_2}$ is the final fv obtained using FFNND_2 method. Feature extraction using the proposed FFNND_2 method is shown graphically in figure 6.2. As shown in figure 6.2, the proposed FFNND_2 method generates two feature codes, as it utilizes two networks. The process of feature extraction using FFNND_2 method is shown in algorithm 6.2. The value of 'L' in algorithm 6.1 is 10.

6.2 Results and Comparison Analysis

For experimental analysis, multi-class SVM employing One vs One (OVO) and One vs All (OVA) approaches with a linear kernel is followed. In this section, the fv length comparison of proposed methods, the experimental results and the comparison of proposed methods with the existing methods is reported.

6.2.1 Feature Vectors Comparison

Although, the proposed FFNND_1 and FFNND_2 methods generate three and two feature codes respectively, the fv length (10) is much lesser when compared to majority of the existing feature descriptors (256). The fv length of FFNND_1 and FFNND_2 methods is 30 and 20 respectively. In order to effectively reduce the fv length, furthermore, the experiments

Algorithm 6.2 Feature Extraction through $FFNND_2$

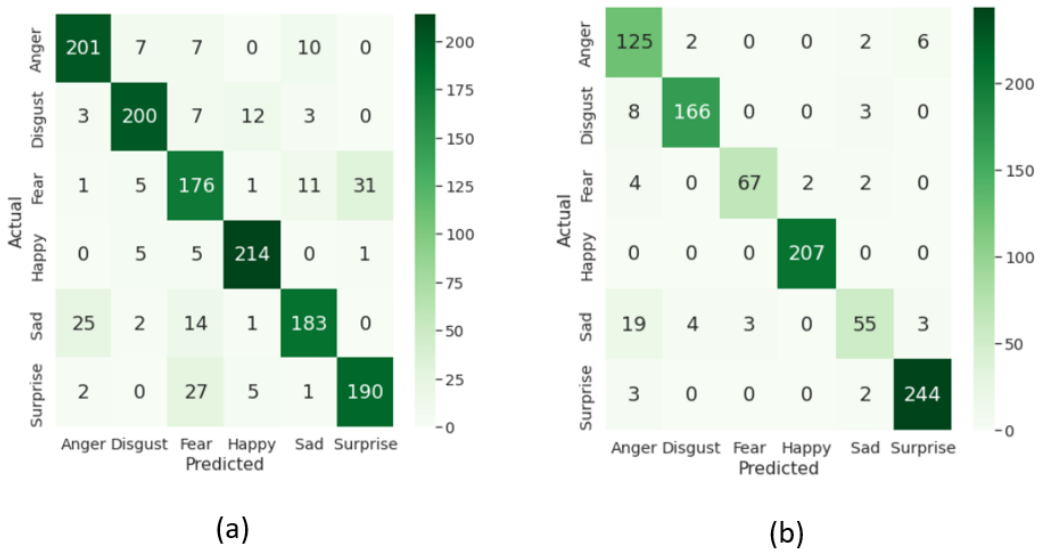
Input: An Input image (Img) of size N x N
Output: Feature vector fv_{ffnnd_2} of size ⌈(N-4)/C⌉ * ⌈(N-4)/C⌉ * 2 * L

procedure $FFNND_2$(Img)
  Initialization: fv_4, $fv_5 \leftarrow \{\ \}$
  Load an input image (Img)
  for all a ∈ range(1,N-4) **do**
    for all b ∈ range(1,N-4) **do**
      Block = img(a:a+4,b:b+4)
      Assign the pixel values to Rook, Bishop and Knight in the 5 x 5 block.
      Calculate fv_4 using eqs.(6.31) to (6.34)
      Calculate fv_5 using eqs.(6.35) to (6.38)
      Normalize fv_4 and fv_5 as shown in eqs.(6.39) and (6.40)
    end for
  end for
  Create two feature response maps obtained from fv_4 and fv_5 by reshaping them to (N-4) x (N-4).
  Each feature response map is partitioned into C x C non-overlapping blocks.
  Histograms are extracted block wise for both the feature response maps using eqs.(6.41) and (6.42).
  Concatenate all the histograms to obtain fv_{ffnnd_2} using eq.(6.43).
  return fv_{ffnnd_2}
end procedure

have been performed with varying block sizes to find out the optimal recognition accuracy.

6.2.2 Experiments for Six Expressions

The experiments for six expressions have been conducted on different 'in the lab' datasets. The proposed FFNND methods have been implemented with logsig and tanh activation functions under varying block sizes with multi-class SVM and the results have been tabulated. In table 6.1, for each dataset, the recognition accuracy comparison of $FFNND_1$ method with logsig activation function under varying block sizes using OVO-SVM classifier is shown. In table 6.2, the recognition accuracy comparison of $FFNND_1$ with logsig activation function under varying block sizes using OVA-SVM classifier is shown. In table 6.3, the recognition accuracy comparison of $FFNND_1$ with tanh activation function under varying block sizes using OVO-SVM classifier is shown. In table 6.4, the recognition accuracy comparison of $FFNND_1$ with tanh activation function under varying block sizes



(a) (b)

Figure 6.3: Confusion matrix for six expressions on (a) MUG dataset using $FFNND_2$ method and (b) CK+ dataset using $FFNND_1$ method

using OVA-SVM classifier is shown. In table 6.5, the recognition accuracy comparison of $FFNND_2$ with logsig activation function under varying block sizes using OVO-SVM classifier is shown. In table 6.6, the recognition accuracy comparison of $FFNND_2$ with logsig activation function under varying block sizes using OVA-SVM classifier is shown. In table 6.7, the recognition accuracy comparison of $FFNND_2$ with tanh activation function under varying block sizes using OVO-SVM classifier is shown. In table 6.8, the recognition accuracy comparison of $FFNND_2$ with tanh activation function under varying block sizes using OVA-SVM classifier is shown.

Among the proposed methods with different block sizes, $FFNND_1$ method with logsig activation function achieved an optimal recognition accuracy of 67.87%, 93.17% and 82.86% on JAFFE, CK+ and KDEF datasets with 9 x 9, 11 x 11 and 9 x 9 block sizes respectively. $FFNND_2$ method with logsig activation function achieved an optimal recognition accuracy of 86.22%, 72.78% and 91.67% on MUG, OULU and ADFES datasets with 8 x 8, 9 x 9 and 8 x 8 block sizes respectively. $FFNND_2$ method with tanh activation function achieved an optimal recognition accuracy of 96.42% and 88.89% on TFEID and WSEFEP datasets with 12 x 12 and 9 x 9 block sizes respectively. The confusion matrix obtained using $FFNND_2$ method for MUG dataset is presented in figure 6.3(a) and for CK+ dataset us-

Table 6.1: Recognition accuracy of $FFNND_1$ method with logsig activation function under varying block sizes for six expressions using OVO-SVM classifier for different datasets

Dataset	**Block Size**								
	8 x 8	**9 x 9**	**10 x 10**	**11 x 11**	**12 x 12**	**13 x 13**	**14 x 14**	**15 x 15**	**16 x 16**
JAFFE	60.61	64.37	63.29	57.52	59.27	63.15	59.09	56.45	54.42
MUG	85.11	81.70	80.37	79.33	81.93	80.15	79.78	80.44	78.22
CK+	90.69	90.90	90.26	93.17	91.89	89.61	90.60	88.87	86.92
OULU	66.87	68.82	70.14	70.90	69.79	70.97	68.40	70.42	70.35
TFEID	94.33	93.42	93.75	94.33	92.5	92.29	90.46	91.58	91.25
KDEF	81.19	81.67	79.52	80.24	77.62	78.1	77.14	78.57	78.81
WSEFEP	87.78	85.56	87.22	86.67	83.89	83.33	83.33	81.67	84.44
ADFES	87.12	88.48	89.39	87.73	85.45	88.64	81.82	87.88	85.61

Table 6.2: Recognition accuracy of $FFNND_1$ method with logsig activation function under varying block sizes for six expressions using OVA-SVM classifier for different datasets

Dataset	**Block Size**								
	8 x 8	**9 x 9**	**10 x 10**	**11 x 11**	**12 x 12**	**13 x 13**	**14 x 14**	**15 x 15**	**16 x 16**
JAFFE	60.08	67.87	63.98	59.93	61.82	59.95	63.20	61.72	55.06
MUG	82.15	81.19	79.04	79.33	78.44	80.07	76.30	76.44	75.63
CK+	89.71	90.81	88.94	92.62	91.23	90.03	89.83	86.03	87.12
OULU	69.38	67.99	67.13	68.75	68.33	67.57	65.56	67.01	68.89
TFEID	93.71	94.67	91.96	94.13	92.20	91.46	90.13	90.25	90.04
KDEF	79.52	82.86	79.05	75.95	74.05	78.81	77.62	76.19	74.05
WSEFEP	86.67	85.56	85	85.56	84.44	80	81.67	76.11	78.33
ADFES	87.88	88.64	87.12	88.48	88.48	88.64	83.33	85.61	83.33

Table 6.3: Recognition accuracy of $FFNND_1$ method with tanh activation function under varying block sizes for six expressions using OVO-SVM classifier for different datasets

Dataset	**Block Size**								
	8 x 8	**9 x 9**	**10 x 10**	**11 x 11**	**12 x 12**	**13 x 13**	**14 x 14**	**15 x 15**	**16 x 16**
JAFFE	57.29	59.31	60.29	57.77	52.86	59.30	57.97	54.33	49.93
MUG	79.33	79.48	79.70	78.44	77.85	78.81	76.81	76.67	74.81
CK+	89.38	90.47	90.99	91.10	90.47	89.17	88.98	89.70	86.35
OULU	67.71	67.5	68.89	69.93	68.13	67.57	65.9	67.78	70.62
TFEID	95.5	94.17	92.75	92.92	92.83	91.58	91.08	89.83	91.67
KDEF	77.14	79.52	76.67	75.95	76.19	75.95	74.52	73.51	72.38
WSEFEP	87.22	83.89	82.78	85.56	83.33	82.78	81.67	81.67	81.11
ADFES	85.61	86.21	87.88	81.67	79.39	84.85	80.30	81.06	79.55

Table 6.4: Recognition accuracy of $FFNND_1$ method with tanh activation function under varying block sizes for six expressions using OVA-SVM classifier for different datasets

Dataset	**Block Size**								
	8 x 8	**9 x 9**	**10 x 10**	**11 x 11**	**12 x 12**	**13 x 13**	**14 x 14**	**15 x 15**	**16 x 16**
JAFFE	56.80	60.50	64.84	62.63	51.96	61.52	61.90	58.73	54.44
MUG	80.22	77.78	78.52	76.07	76.15	77.70	73.48	76.07	75.04
CK+	89.60	90.57	89.37	90.23	89.29	89.49	89.51	88.54	86.69
OULU	67.01	68.61	67.43	68.26	68.61	67.99	67.85	65.49	66.67
TFEID	95.29	95.5	92.92	94.08	93.67	94.58	91.58	90.46	91.46
KDEF	75.24	78.33	75.48	72.38	74.52	76.67	71.24	72.86	72.38
WSEFEP	87.22	84.44	83.89	85.56	81.67	78.89	78.89	77.78	78.33
ADFES	88.64	89.39	85.61	88.64	81.67	81.82	79.55	80.30	83.33

Table 6.5: Recognition accuracy of $FFNND_2$ method with logsig activation function under varying block sizes for six expressions using OVO-SVM classifier for different datasets

Dataset	**Block Size**								
	8 x 8	**9 x 9**	**10 x 10**	**11 x 11**	**12 x 12**	**13 x 13**	**14 x 14**	**15 x 15**	**16 x 16**
JAFFE	65.07	64.98	65.40	60.65	65.48	62.66	63.49	64.39	62.76
MUG	84.44	84.30	83.85	82.52	82.81	83.93	80.89	80.15	81.04
CK+	91.03	90.82	89.29	90.91	91.15	90.26	88.98	89.20	88.65
OULU	71,53	72.78	71.81	72.43	72.29	71.74	72.43	71.67	68.75
TFEID	94.67	95	94.25	94.67	93.83	93	94	92.5	92.92
KDEF	81.67	77.14	79.52	80.24	79.76	77.62	76.62	74.05	76.67
WSEFEP	84.44	85.56	86.11	83.33	82.22	85	85	83.33	81.11
ADFES	86.21	90.91	87.73	89.24	86.36	89.09	85.45	87.88	86.97

Table 6.6: Recognition accuracy of $FFNND_2$ method with logsig activation function under varying block sizes for six expressions using OVA-SVM classifier for different datasets

Dataset	**Block Size**								
	8 x 8	**9 x 9**	**10 x 10**	**11 x 11**	**12 x 12**	**13 x 13**	**14 x 14**	**15 x 15**	**16 x 16**
JAFFE	62.66	65.42	65.54	62.61	63.62	58.91	63.73	58.43	63.67
MUG	86.22	83.56	83.41	83.48	80.52	80.96	79.26	75.04	77.63
CK+	89.93	91.55	88.54	90.48	91.57	89.71	87.89	88.64	88.65
OULU	69.65	70.15	70.76	69.93	70.49	68.75	70	66.94	65.63
TFEID	93.67	94.58	95	93.25	93.42	95.17	94.42	92.17	92.92
KDEF	81.90	79.76	80.24	80.24	76.9	74.46	72.62	74.29	73.57
WSEFEP	85.56	86.67	88.33	83.89	84.44	86.67	84.44	85.56	80
ADFES	91.67	89.39	90.15	89.39	87.88	87.73	84.09	87.12	86.21

Table 6.7: Recognition accuracy of $FFNND_2$ method with tanh activation function under varying block sizes for six expressions using OVO-SVM classifier for different datasets

Dataset	**Block Size**								
	8 x 8	**9 x 9**	**10 x 10**	**11 x 11**	**12 x 12**	**13 x 13**	**14 x 14**	**15 x 15**	**16 x 16**
JAFFE	61.18	60	58.24	57.05	59.98	56.21	60.36	57.37	56.69
MUG	83.04	83.78	84.67	81.19	82.81	81.70	79.93	79.11	78.52
CK+	92.42	90.58	89.83	91.01	89.94	91.02	87.91	87.01	87.02
OULU	69.79	70.76	71.39	71.74	71.25	71.11	70.21	71.25	69.72
TFEID	95.17	94.5	93.67	93.17	95	92.92	90.5	89.83	92.58
KDEF	79,05	78,33	77.86	76.9	75	76.19	75	74.29	74.28
WSEFEP	85.56	86.67	86.11	86.11	82.78	85.56	84.44	83.33	84.44
ADFES	89.39	87.88	88.64	87.73	89.39	85.61	83.18	86.21	82.42

Table 6.8: Recognition accuracy of $FFNND_2$ method with tanh activation function under varying block sizes for six expressions using OVA-SVM classifier for different datasets

Dataset	**Block Size**								
	8 x 8	**9 x 9**	**10 x 10**	**11 x 11**	**12 x 12**	**13 x 13**	**14 x 14**	**15 x 15**	**16 x 16**
JAFFE	58.01	64.05	57.18	59.19	57.96	59.69	59.86	59.37	52.17
MUG	82.74	84.15	83.85	80.59	81.78	80.59	78.67	79.33	74.22
CK+	89.92	89.61	88.52	89.50	87.89	90.47	86.82	88.53	87.45
OULU	69.24	69.72	68.82	70.42	68.54	69.79	69.1	68.02	66.88
TFEID	94.46	95.42	94.38	94.17	96.42	93.42	91.42	91.58	94.25
KDEF	80	78.1	77.38	76.67	76.43	75.95	72.86	74.29	73.1
WSEFEP	88.33	88.89	87.78	87.78	83.33	84.44	84.44	83.89	82.78
ADFES	90.15	89.39	90.15	87.12	84.70	87.12	87.88	87.12	85.45

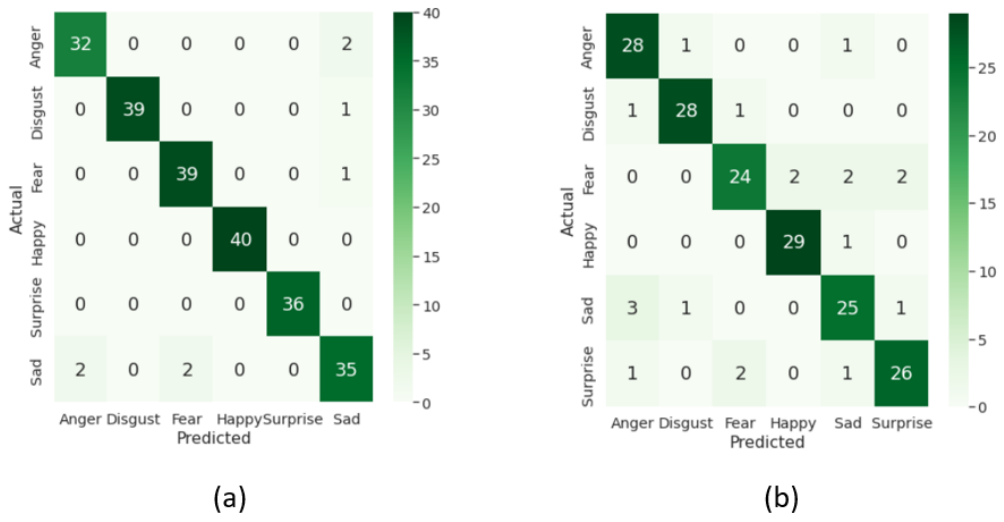


Figure 6.4: Confusion matrix for six expressions on (a) TFEID dataset using $FFNND_2$ method and (b) WSEFEP dataset using $FFNND_2$ method

ing $FFNND_1$ method is presented in figure 6.3(b). The confusion matrix obtained using $FFNND_2$ method for TFEID dataset is presented in figure 6.4(a) and for WSEFEP dataset using $FFNND_2$ method is presented in figure 6.4(b). In table 6.9, the comparison analysis of the proposed methods with the existing variants of binary patterns, implemented in our environment setup is shown. In table 6.10, the comparison analysis of the proposed methods with the existing methods is shown. In tables 6.9 and 6.10, the proposed methods and their recognition accuracy has been highlighted in bold. The comparison analysis of proposed method with the existing variants of binary patterns on CK+ and TFEID datasets is shown in figure 6.5. From table 6.9, the proposed FFNND methods achieved better recognition accuracy than existing variants of binary patterns for all other datasets except OULU and KDEF datasets. From table 6.10, FFNND methods achieved better recognition accuracy than existing methods for JAFFE, CK+, TFEID, KDEF, WSEFEP and ADFES datasets. In case of MUG dataset, HiNet and ResNet50 methods achieved 1.58% and 0.66% more than the proposed $FFNND_2$ method. Although, the other variants of binary patterns and existing methods achieved better recognition accuracy than the proposed FFNND methods in case of OULU dataset, the proposed methods are simple, easily implementable and have lesser fv length when compared to majority of the existing methods.

Table 6.9: Comparison analysis with existing variants of binary patterns for six expressions for different 'in the lab' datasets

Method	JAFFE	MUG	CK+	OULU	TFEID	KDEF	WSEFEP	ADFES
LBP [46]	56.66	82.65	89.97	75.34	92.42	80.95	87.22	90.16
LDP [97]	52.77	82.87	90.84	72.43	93.67	80.95	86.67	90.91
LDN [31]	56.66	81.96	88.84	72.29	93.33	82.62	87.22	88.64
CSLBP [28]	53.28	83.94	90.68	71.53	94.25	81.67	84.44	87.12
LGC [84]	58.89	86.22	89.61	73.47	95.00	83.81	87.78	90.91
LDTP [32]	55.55	82.04	89.60	72.29	93.25	83.57	87.78	88.64
LDTeRP [33]	58.54	80.15	85.89	68.54	90.75	81.43	81.11	84.89
ALDP [35]	55.09	82.89	89.06	70.28	93.54	80.95	85.00	85.61
MSBP [88]	56.75	85.78	90.58	73.41	94.25	83.10	86.67	90.15
LDSP [34]	56.70	85.19	91.54	68.47	94.50	82.38	85.00	86.36
LDDSCP [99]	59.55	86.15	89.61	73.47	95.50	84.76	83.89	89.39
RADAP [17]	57.22	83.48	90.63	75.90	94.17	82.38	87.78	91.67
LBP + LNeP [134]	61.29	86.52	92.21	75.00	93	83.57	88.89	90.15
$FFNND_1$	**67.87**	**85.11**	**93.17**	**70.97**	**95.50**	**82.86**	**87.78**	**89.39**
$FFNND_2$	**65.54**	**86.22**	**92.42**	**72.78**	**96.42**	**81.90**	**88.89**	**91.67**

Table 6.10: Comparison with existing methods for six expressions for different 'in the lab' datasets

Dataset	Method	Accuracy	Dataset	Method	Accuracy
JAFFFE	IFRBC [86]	62.29	MUG	ResNet50 [17]	86.88
	ResNet50 [17]	59.44		DAGSVM [129]	82.28
	LOOP [11]	58.72		LOOP [11]	85.33
	WLGC-HD [57]	60.7		HiNet [108]	87.8
	FFNND$_1$	**67.87**		**FFNND$_1$**	**85.11**
	FFNND$_2$	**65.54**		**FFNND$_2$**	**86.22**
CK+	ResNet50 [17]	89.32	OULU	ResNet50 [17]	73.1
	HiNet [108]	91.40		HiNet [108]	74.3
	WLGC-HD [57]	72.80		VGG16 [17]	73.4
	FFNND$_1$	**93.17**		**FFNND$_1$**	**70.97**
	FFNND$_2$	**92.42**		**FFNND$_2$**	**72.78**
TFEID	DSNGE [85]	93.89	KDEF	IFRBC [86]	77.98
	DAMCNN [102]	93.65		ICVR [86]	76.31
	LIoP + HOG [123]	93.50		HOG [108]	82.17
	FFNND$_1$	**95.50**		**FFNND$_1$**	**82.86**
	FFNND$_2$	**96.42**		**FFNND$_2$**	**81.90**
WSEFEP	LOOP [11]	87.78	ADFES	LOOP [11]	91.67
	FFNND$_1$	**87.78**		**FFNND$_1$**	**88.89**
	FFNND$_2$	**88.89**		**FFNND$_2$**	**91.67**

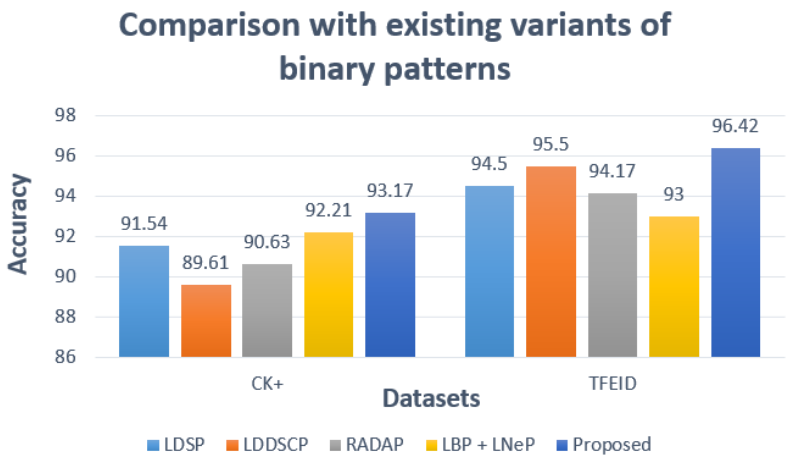


Figure 6.5: Comparison analysis of proposed method with existing variants of binary patterns for six expressions on CK+ and TFEID datasets

6.2.3 Experiments for Seven Expressions

The experiments for seven expressions have been conducted on different FER datasets using FFNND methods. The proposed FFNND methods have been implemented with logsig and tanh activation functions under varying block sizes with multi-class SVM and the results have been tabulated for seven expressions. In table 6.11, for each dataset, the recognition accuracy comparison of $FFNND_1$ method with logsig activation function under varying block sizes using OVO-SVM classifier is shown. In table 6.12, the recognition accuracy comparison of $FFNND_1$ with logsig activation function under varying block sizes using OVA-SVM classifier is shown. In table 6.13, the recognition accuracy comparison of $FFNND_1$ with tanh activation function under varying block sizes using OVO-SVM classifier is shown. In table 6.14, the recognition accuracy comparison of $FFNND_1$ with tanh activation function under varying block sizes using OVA-SVM classifier is shown. In table 6.15, the recognition accuracy comparison of $FFNND_2$ with logsig activation function under varying block sizes using OVO-SVM classifier is shown. In table 6.16, the recognition accuracy comparison of $FFNND_2$ with logsig activation function under varying block sizes using OVA-SVM classifier is shown. In table 6.17, the recognition accuracy comparison of $FFNND_2$ with tanh activation function under varying block sizes using OVO-SVM classifier is shown. In table 6.18, the recognition accuracy comparison of $FFNND_2$ with tanh

activation function under varying block sizes using OVA-SVM classifier is shown.

Among the proposed methods with different block sizes, $FFNND_1$ method with a block size of 9 x 9 achieved an optimal recognition accuracy of 66.14% on JAFFE dataset. $FFNND_2$ method with a block size of 8 x 8 achieved an optimal recognition accuracy of 87.27%, 80.91%, 85.71%, 92.21% and 99.83% on CK+, KDEF, WSEFEP, ADFES and FERG datasets respectively. $FFNND_2$ method with a block size of 9 x 9 achieved an optimal recognition accuracy of 81.14% and 70.24% on MUG and OULU datasets respectively. $FFNND_2$ method with a block size of 10 x 10 and 13 x 13 achieved an optimal recognition accuracy of 95.48% and 67.44% on TFEID and RAF datasets respectively. The confusion matrix obtained using $FFNND_1$ method for JAFFE dataset is presented in figure 6.6(a) and for ADFES dataset using $FFNND_2$ method is presented in figure 6.6(b).

The confusion matrix obtained using $FFNND_2$ for RAF dataset is presented in figure 6.7(a) and for FERG dataset using $FFNND_2$ method is presented in figure 6.7(b). In table 6.19, the comparison analysis of the proposed methods with the existing variants of binary patterns is shown. In table 6.20, the comparison analysis of the proposed methods with the existing methods is shown. In tables 6.19 and 6.20, the proposed methods and their recognition accuracy has been highlighted in bold. The comparison analysis of proposed method with the existing variants of binary patterns on TFEID and ADFES datasets is shown in figure 6.8. From table 6.19, the proposed FFNND methods achieved better recognition accuracy than the existing variants of binary patterns in case of JAFFE, MUG, CK+, TFEID, KDEF and ADFES datasets. In case of OULU and WSEFEP datasets, RADAP method achieved 4.1% and 1.71% better recognition accuracy than the proposed $FFNND_2$ method. Although, RADAP method achieved better recognition accuracy, the fv length of RADAP method is very high when compared to FFNND methods. From table 6.20, the proposed FFNND methods achieved better recognition accuracy on JAFFE, TFEID, WSEFEP, ADFES and FERG datasets.

From table 6.20, in case of MUG dataset, HiNet and ResNet50 methods achieved 6.06% and 4.44% more than the proposed $FFNND_2$ method. In case of OULU dataset, HiNet and VGG19 methods achieved 1.76% and 0.26% better recognition accuracy than the proposed $FFNND_2$ method. In case of KDEF dataset, SAFL method achieved 0.31% better recog-

Table 6.11: Recognition accuracy of $FFNND_1$ method with logsig activation function under varying block sizes for seven expressions using OVO-SVM classifier for different datasets

Dataset	**Block Size**								
	8 x 8	**9 x 9**	**10 x 10**	**11 x 11**	**12 x 12**	**13 x 13**	**14 x 14**	**15 x 15**	**16 x 16**
JAFFE	59.69	63.87	62.88	58.02	59.19	60.46	58.71	53.21	53.12
MUG	79.05	76.63	75.75	73.52	76.89	73.71	74.60	74.03	70.10
CK+	85.03	83.91	83.91	83.14	83.59	82.90	83.08	82.59	81.31
OULU	68.93	67.98	68.51	67.86	68.63	68.21	66.43	65.77	68.33
TFEID	93.33	93.27	92.14	93.33	91.79	92.92	92.92	91.07	90
KDEF	78.37	79.80	76.94	78.16	76.53	74.90	74.49	75.71	75.92
WSEFEP	83.17	81.43	82.38	83.25	82.30	77.46	78.97	78.41	80.32
ADFES	87.01	88.85	87.66	88.10	86.26	87.66	81.82	82.47	83.12
RAF	65.32	63.53	61.70	61.38	58.96	61.38	61.51	60.76	59.88
FERG	98.11	98.13	98.14	98.11	97.94	97.69	97.83	97.93	97.43

Table 6.12: Recognition accuracy of $FFNND_1$ method with logsig activation function under varying block sizes for seven expressions using OVA-SVM classifier for different datasets

Dataset	**Block Size**								
	8 x 8	**9 x 9**	**10 x 10**	**11 x 11**	**12 x 12**	**13 x 13**	**14 x 14**	**15 x 15**	**16 x 16**
JAFFE	58.83	66.14	60.42	57.53	61.63	54.44	61.15	54.48	51.90
MUG	77.21	75.62	73.14	76	75.62	75.56	71.24	71.43	70.98
CK+	85.66	85.45	83.29	85.33	84.31	82.87	82.97	80.26	81.24
OULU	68.15	69.11	67.02	66.13	64.82	65.42	63.87	62.62	64.88
TFEID	93.55	95.12	92.06	93.11	91.82	92.24	90.87	90.24	90.04
KDEF	78,98	80.41	77.96	75.51	72.86	74.69	73.47	72.92	72.65
WSEFEP	81.35	82.86	80.48	81.90	82.38	76.59	77.14	74.21	76.59
ADFES	87.66	87.66	87.55	88.20	86.90	88.96	81.71	79.22	81.82
RAF	57.27	59.16	59.91	61.44	60.07	63.14	64.99	63.04	62.68
FERG	98.13	98.11	98.13	97.99	97.77	97.59	97.34	97.14	95.90

Table 6.13: Recognition accuracy of $FFNND_1$ method with tanh activation function under varying block sizes for seven expressions using OVO-SVM classifier for different datasets

Dataset	**Block Size**								
	8 x 8	**9 x 9**	**10 x 10**	**11 x 11**	**12 x 12**	**13 x 13**	**14 x 14**	**15 x 15**	**16 x 16**
JAFFE	54.78	61.33	57.88	55.77	50.80	60.85	58.82	49.07	44.22
MUG	73.40	73.02	74.16	70.92	72.13	72.19	68.06	71.81	68.06
CK+	83.08	83.70	83.07	83.24	82.86	82.18	82.19	81.39	78.50
OULU	64.71	67.02	66.43	66.67	66.31	65.89	64.35	65	67.14
TFEID	93.57	91.37	90.57	89.46	91.31	89.23	90.60	89.17	89.29
KDEF	75.71	77.96	73.67	74.08	75.10	75.10	72.65	70	72.04
WSEFEP	80.71	76.59	79.92	80.32	79.37	74.76	78.41	76.98	76.11
ADFES	83.77	84.96	85.06	84.31	79.11	77.92	76.62	75.97	74.68
RAF	64.96	64.08	62.32	61.90	60.23	60.69	57.79	56.71	54.92
FERG	98.14	98.14	98.13	98.13	98.01	98.09	98.06	98.03	97.97

Table 6.14: Recognition accuracy of $FFNND_1$ method with tanh activation function under varying block sizes for seven expressions using OVA-SVM classifier for different datasets

Dataset	**Block Size**								
	8 x 8	**9 x 9**	**10 x 10**	**11 x 11**	**12 x 12**	**13 x 13**	**14 x 14**	**15 x 15**	**16 x 16**
JAFFE	53.45	60.41	62.77	61.79	50.80	59.84	62.52	56.09	46.72
MUG	75.24	71.68	74.35	72.83	70.03	72.19	66.86	69.59	69.52
CK+	84.20	84.58	83.30	84.20	81.49	82.17	81.61	81.11	80.25
OULU	63.45	66.96	64.05	63.45	65.24	65.36	62.62	61.61	62.60
TFEID	94.14	93.93	92.02	91.61	92.02	91.64	89.70	88.19	90.18
KDEF	74.49	75.71	74.49	71.63	74.08	73.67	70.2	67.96	70.61
WSEFEP	85.63	78.02	78.49	79.44	77.14	77.06	76.19	73.33	76.51
ADFES	87.66	88.31	85.71	89.61	85.61	83.77	77.27	77.16	80.52
RAF	58.18	55.02	54.04	54.63	54.92	55.83	56.65	57.27	55.71
FERG	98.14	98.14	98.09	98.04	97.97	97.97	97.94	98	97.94

Table 6.15: Recognition accuracy of $FFNND_2$ method with logsig activation function under varying block sizes for seven expressions using OVO-SVM classifier for different datasets

Dataset	**Block Size**								
	8 x 8	**9 x 9**	**10 x 10**	**11 x 11**	**12 x 12**	**13 x 13**	**14 x 14**	**15 x 15**	**16 x 16**
JAFFE	63.27	64.34	64.66	58.64	61.89	60.41	60.60	59	60.22
MUG	79.49	81.14	77.59	77.90	78.48	77.59	75.05	74.41	74.16
CK+	85.33	86.20	85.58	85.76	84.57	85.38	82.79	83.01	82.49
OULU	68.39	70.24	69.52	69.4	69.94	69.46	69.11	68.51	66.01
TFEID	92.86	93.21	93.21	94.35	92.86	92.5	93.45	92.62	93.63
KDEF	79.39	75.92	79.39	78.57	78.98	76.73	73.67	72.65	75.51
WSEFEP	83.25	83.25	83.73	79.92	80.89	80.87	80.40	79.37	78.02
ADFES	86.36	89.50	87.01	87.55	87.01	86.80	83.01	85.71	86.26
RAF	64.24	62.78	63.33	63.62	63.85	64.05	64.54	63.14	64.02
FERG	98.20	98.14	98.24	97.96	98.36	98	98.31	99.36	99

Table 6.16: Recognition accuracy of $FFNND_2$ method with logsig activation function under varying block sizes for seven expressions using OVA-SVM classifier for different datasets

Dataset	**Block Size**								
	8 x 8	**9 x 9**	**10 x 10**	**11 x 11**	**12 x 12**	**13 x 13**	**14 x 14**	**15 x 15**	**16 x 16**
JAFFE	60.28	62.71	62.73	60.78	59.39	59.89	59.95	54.71	62.22
MUG	80.25	78.92	76.89	78.10	76.06	73.40	74.10	69.52	70.60
CK+	86.48	86.77	84.48	84.78	84.65	85.99	81.89	82.33	80.50
OULU	67.02	68.15	67.20	66.13	65.36	67.38	65.38	63.27	60.60
TFEID	93.51	93.99	95.48	93.63	90.60	92.92	93.57	91.61	91.79
KDEF	80.91	78.16	79.59	77.96	75.10	72.86	71.02	71.43	70.82
WSEFEP	85.71	82.30	81.90	80.87	80.95	80.40	80	80.32	73.73
ADFES	92.21	88.96	90.26	88.96	87.66	86.90	82.47	83.12	84.31
RAF	62.32	62.19	64.28	64.80	66.75	67.44	66.26	65.55	64.02
FERG	98.11	98.06	97.96	97.59	97.54	97.80	97.64	96.36	96.93

Table 6.17: Recognition accuracy of $FFNND_2$ method with tanh activation function under varying block sizes for seven expressions using OVO-SVM classifier for different datasets

Dataset	**Block Size**								
	8 x 8	**9 x 9**	**10 x 10**	**11 x 11**	**12 x 12**	**13 x 13**	**14 x 14**	**15 x 15**	**16 x 16**
JAFFE	56.78	60.92	56.57	52.87	55.92	53.26	58.92	54.37	52.55
MUG	77.59	80.44	78.86	77.46	78.22	76.51	74.73	74.29	70.03
CK+	84.65	84.78	84.35	84.79	84.86	84.57	80.79	80.87	79.85
OULU	67.20	67.38	66.61	67.02	65.18	66.49	66.01	63.04	63.04
TFEID	94.40	91.37	92.50	91.43	93.93	90.95	90.95	89.11	92.08
KDEF	77.55	76.12	77.14	76.73	74.08	74.08	71.22	72.86	73.88
WSEFEP	82.70	81.75	81.75	83.65	79.44	80.79	80.87	79.84	77.06
ADFES	88.96	88.96	88.31	87.55	87.66	83.66	83.12	85.61	81.07
RAF	65.12	64.05	62.94	61.44	58.51	64.05	60.79	57.66	59.19
FERG	98.23	98.13	98.14	98.09	99.74	98.13	98.07	98.79	98.11

Table 6.18: Recognition accuracy of $FFNND_2$ method with tanh activation function under varying block sizes for seven expressions using OVA-SVM classifier for different datasets

Dataset	**Block Size**								
	8 x 8	**9 x 9**	**10 x 10**	**11 x 11**	**12 x 12**	**13 x 13**	**14 x 14**	**15 x 15**	**16 x 16**
	57.94	56.45	57.07	54.74	56	59.40	52.97	54.37	50.53
MUG	78.10	78.22	78.79	77.27	77.08	74.03	73.21	71.56	67.37
CK+	87.27	86.24	84.99	84.69	83.48	84.43	80.45	79.31	78.90
OULU	67.26	68.04	68.10	68.04	69.10	67.26	66.67	65.83	66.19
TFEID	93.13	93.57	93.49	92.86	93.87	91.73	89.26	91.37	92.92
KDEF	77.22	76.53	75.31	75.51	75.71	72.45	71.22	71.84	70.20
WSEFEP	85.16	84.21	83.65	81.43	80.95	79.37	78.02	76.51	78.10
ADFES	89.61	88.31	90.91	88.96	82.47	84.31	85.71	82.47	82.47
RAF	55.22	57.95	59.13	58.74	61.60	59.88	62.71	61.21	62.06
FERG	99.83	98.14	98.13	98.09	98.64	98.19	97.94	98.36	97.94

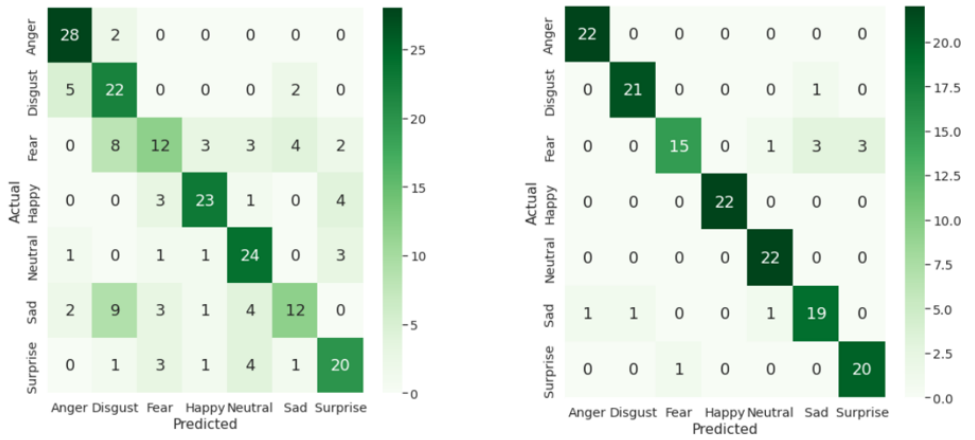


Figure 6.6: Confusion matrix for seven expressions on (a) JAFFE dataset using $FFNND_1$ method and (b) ADFES dataset using $FFNND_2$ method

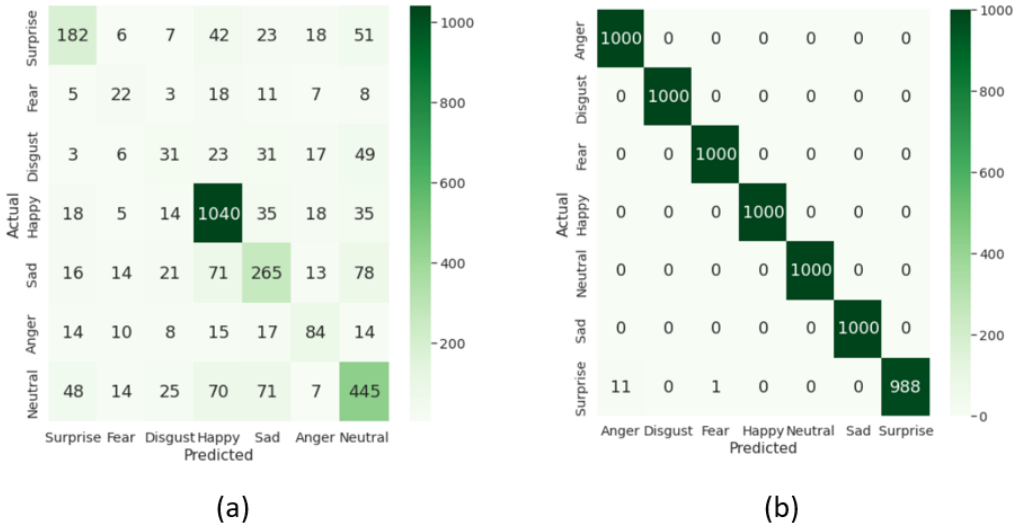


Figure 6.7: Confusion matrix for seven expressions on (a) RAF dataset using $FFNND_2$ method and (b) FERG dataset using $FFNND_2$ method

Table 6.19: Comparison analysis with existing variants of binary patterns for seven expressions for different 'in the lab' datasets

Method	JAFFE	MUG	CK+	OULU	TFEID	KDEF	WSEFEP	ADFES
LBP [46]	53.65	76.16	83.96	65.71	92.02	78.16	82.38	87.66
LDP [97]	52.10	78.70	84.80	69.16	86.20	80.61	80.95	90.26
LDN [31]	54.87	77.85	83.35	70.89	93.51	80.75	83.81	91.56
CSLBP [28]	52.40	79.37	85.51	57.98	89.44	80.20	79.44	86.36
LGC [84]	57.59	82.03	86.71	71.13	93.43	81.02	84.29	90.91
LDTP [32]	51.32	78.70	83.08	68.86	93.15	80.81	85.71	85.71
LDTeRP [33]	51.70	78.11	81.40	64.53	90.18	77.35	79.52	79.87
ALDP [35]	51.53	77.59	85.61	67.74	92.36	76.53	80.48	83.77
MSBP [88]	56.81	81.40	86.04	73.75	93.27	81.22	83.73	91.56
LDSP [34]	52.49	80.63	84.19	63.81	93.15	80.61	85.00	85.06
LDDSCP [99]	57.37	80.95	84.93	69.57	90.95	81.84	82.38	88.96
RADAP [17]	56.20	80.26	84.60	74.34	93.27	80.20	87.42	90.91
LBP + LNeP [134]	59.04	81.45	86.30	73.20	93.27	81.84	86.19	90.26
$FFNND_1$	**66.14**	**79.05**	**85.66**	**69.11**	**95.12**	**80.41**	**85.63**	**89.61**
$FFNND_2$	**64.66**	**81.14**	**87.27**	**70.24**	**95.48**	**80.91**	**85.71**	**92.21**

Table 6.20: Comparison with existing methods for seven expressions for different datasets

Dataset	Method	Accuracy	Dataset	Method	Accuracy
JAFFE	PCANet [117]	58.35	MUG	CBA [90]	78.57
	ResNet50 [17]	57.13		ResNet50 [17]	85.58
	LOOP [11]	59.67		LOOP [11]	79.68
	WLGC-HD [57]	58.2		HiNet [108]	87.2
	FFNND$_1$	**66.14**		**FFNND $_1$**	**79.05**
	FFNND$_2$	**64.66**		**FFNND $_2$**	**81.14**
CK+	ResNet50 [17]	87.31	OULU	ResNet50 [17]	65.4
	HiNet [108]	88.6		HiNet [108]	72
	DLFS [107]	83.72		VGG19 [17]	70.5
	FFNND$_1$	**85.66**		**FFNND$_1$**	**69.11**
	FFNND$_2$	**87.27**		**FFNND$_2$**	**70.24**
TFEID	DAMCNN [102]	93.36	KDEF	DLFS [107]	78.60
	Pyramid+SBDT [87]	93.38		PCANet [117]	69.59
	MSDV [132]	93.50		SAFL [108]	81.22
	FFNND$_1$	**95.12**		**FFNND$_1$**	**80.41**
	FFNND$_2$	**95.48**		**FFNND$_2$**	**80.91**
WSEFEP	LOOP[11]	84.68	ADFES	AFM [103]	92.70
	FFNND$_1$	**85.63**		**FFNND$_1$**	**89.61**
	FFNND$_2$	**85.71**		**FFNND$_2$**	**92.21**
RAF	DLPCNN [55]	74.20	FERG	Deep Expr [56]	89.02
	ICID Fusion [101]	75.40		Ensemble Multi-Feature [17]	97
	Sadeghi et al. [93]	76.23		Adversarial NN [105]	98.2
	DCNN+RLPS [110]	72.84		Deep Emotion [106]	99.3
	IFSL [120]	76.9		LBP-AW [16]	96.7
	FFNND$_1$	**65.32**		**FFNND$_1$**	**98.14**
	FFNND$_2$	**67.44**		**FFNND$_2$**	**99.83**

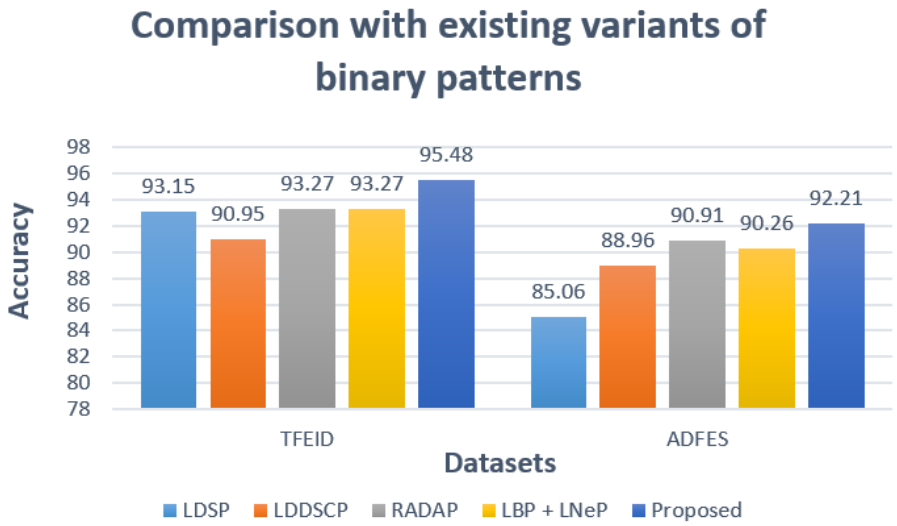


Figure 6.8: Comparison analysis of proposed method with existing variants of binary patterns for seven expressions on TFEID and ADFES datasets

nition than the proposed $FFNND_2$ method. From table 6.20, in case of RAF dataset, the proposed $FFNND_2$ method with logsig activation function achieved an optimal recognition accuracy of 67.44%. The existing deep learning methods achieved better recognition accuracy than the proposed methods. This is because, RAF dataset has images captured from real world under un-constrained environment. Although, some of the existing methods achieved better recongition accuracy than the proposed FFNND methods, the proposed are simple and extracts salient features in a local neighborhood with lesser fv length. In case of FERG dataset, from table 6.20, $FFNND_2$ method outperformed the existing methods.

6.2.4 Experiments for Eight Expressions

The proposed FFNND methods have been implemented with varying block sizes and the results have been reported in table 6.21. Among the proposed methods, $FFNND_2$ method with tanh activation function and 8 x 8 block size achieved an optimal recognition accuracy of 92.79%. The comparison analysis of the proposed methods with the existing variants of binary patterns is reported in the second column of table 6.23. In table 6.23, the proposed methods and their recognition accuracy has been highlighted in bold. From table 6.23, the experimental results indicate that the proposed $FFNND_2$ method outperformed the existing variants of binary patterns in terms of recognition accuracy.

Table 6.21: Recognition accuracy comparison using FFNND methods for eight expressions on TFEID Dataset

Method	**Activation function**	**Classifier**	**Block Size**								
			8 x 8	**9 x 9**	**10 x 10**	**11 x 11**	**12 x 12**	**13 x 13**	**14 x 14**	**15 x 15**	**16 x 16**
$FFNND_1$	logsig	OVO-SVM	88.89	92.04	89.82	89.43	91.04	88.37	90.26	89.19	87.52
		OVA-SVM	88.97	90.40	90.52	87.59	88.90	87.06	89.49	84.68	82.06
	tanh	OVO-SVM	88.45	88.74	88.65	87.07	87.98	85.82	86.83	82.92	82.85
		OVA-SVM	89.98	91.42	89.44	87.47	89.18	82.05	86.78	84.69	81.68
$FFNND_2$	logsig	OVO-SVM	92.79	92.71	92.77	91.03	91.98	86.75	89.75	84.29	88.94
		OVA-SVM	89.65	91.66	90.33	88.29	85.94	82.79	87.32	84.22	83.71
	tanh	OVO-SVM	92.67	91.41	89.60	89.31	86.59	84.99	86.07	85.62	83.87
		OVA-SVM	92.46	92.29	91.45	89.16	90.14	87.66	90.16	86.67	85.43

Table 6.22: Recognition accuracy comparison using FFNND methods for ten expressions on ADFES Dataset

Method	**Activation function**	**Classifier**	**Block Size**								
			8 x 8	**9 x 9**	**10 x 10**	**11 x 11**	**12 x 12**	**13 x 13**	**14 x 14**	**15 x 15**	**16 x 16**
$FFNND_1$	logsig	OVO-SVM	81.83	75.87	79.75	75.92	78.38	78.33	73.19	74.10	72.79
		OVA-SVM	81.01	78.09	79.29	78.15	79.81	79.29	74.10	71.83	77.03
	tanh	OVO-SVM	79.10	77.53	75.87	74.81	72.24	70.61	68.04	68.64	69.85
		OVA-SVM	78.64	79.15	76.32	80.32	76.88	76.06	70.46	65.36	68.94
$FFNND_2$	logsig	OVO-SVM	78.64	78.90	78.65	79.86	79.19	80.41	73.90	75.51	74.86
		OVA-SVM	81.57	79.19	82.02	79.19	80.10	77.94	74.50	74.36	69.87
	tanh	OVO-SVM	77.27	80.25	77.48	80.31	74.95	74.45	73.18	73.59	74.91
		OVA-SVM	80.82	80.56	81.11	75.53	77.88	76.67	74.81	75.41	73.55

Table 6.23: Comparison analysis with existing variants of binary patterns for eight and ten expressions on TFEID and ADFES datasets

Method	Eight Expressions	Ten Expressions
LBP [46]	90.95	83.54
LDP [97]	91.17	85.25
LDN [31]	88.06	85.20
CSLBP [28]	89.45	80.61
LGC [84]	89.95	85.20
LDTP [32]	86.29	82.68
LDTerP [33]	88.59	76.53
ALDP [35]	90.36	78.40
MSBP [88]	89.54	87.17
LDSP [34]	86.68	79.91
LDDSCP [99]	87.66	84.96
RADAP [17]	90.56	84.75
LBP + LNeP [134]	90.81	87.07
$FFNND_1$	**92.04**	**81.83**
$FFNND_2$	**92.79**	**82.02**

6.2.5 Experiments for Ten Expressions

The proposed FFNND methods have been implemented with varying block sizes and the results have been reported in table 6.22. Among the proposed methods, $FFNND_2$ method with logsig activation function and 10 x 10 block size achieved an optimal recognition accuracy of 82.02%. The comparison analysis of the proposed methods with the existing variants of binary patterns is reported in the third column of table 6.23. From table 6.23, the proposed method outperformed recent LDSP method by 2.1% in terms of recognition accuracy. The methods such as MSBP, LDDSCP, RADAP and LBP+LNeP achieved 5.15%, 2.94%, 2.73% and 5.05% more than the proposed $FFNND_2$ method, at the cost of increased fv length.

6.3 Summary

The method employed for feature extraction plays a major role in determining the performance of an FER system. Novel local texture based feature descriptors FFNND ($FFNND_1$

and $FFNND_2$) inspired by the structure of a feed forward neural network have been proposed in this Chapter. The main objective of the proposed methods is to extract salient features in a local neighborhood. $FFNND_1$ extracts three feature codes by capturing the adjacent pixel relationship based on multi-distance information as like LMeP, whereas, $FFNND_2$ extracts two feature codes by capturing the relationship between the pixels located at a radius (rd=2), as like MTP. In this Chapter, the experiments have been conducted with different block sizes using both OVO-SVM and OVA-SVM classifiers for determining the optimal recognition accuracy. From the experimental results, an observation has been made that proposed methods outperformed the standard existing methods in majority of the datasets. In cases of OULU, KDEF and RAF datasets, the proposed methods could not achieve the best recognition accuracy when compared to existing methods. But, the main advantage of the proposed method is that the fv length of proposed FFNND methods is much lesser when compared to majority of the existing feature descriptors.

Chapter 7

Conclusion and Future Scope

7.1 Conclusion

Facial expressions are extremely important in the social interaction as they can display the internal emotions and intentions of an individual. The main task in FER systems is to develop feature descriptors that could effectively classify the facial expressions into various categories. Hence, in this thesis, some local texture based feature descriptors have been presented for FER systems, intended to improve the overall recognition accuracy.

In chapter 3, inspired by the chess game rules, CP, kTP and KTP feature descriptors have been presented for extracting the facial features in a local neighborhood. CP has been proposed with an intention to generate different feature codes for corner, edge and flat portions of an image. Inspired by the Knight tour problem, kTP and KTP feature descriptors have been proposed, which utilizes Knight moves for generating the features by comparing the neighboring pixels with the pixel p_c in a local neighborhood. Local texture based feature descriptors named RMP, RCP, CSP and RCSP have been presented in chapter 4. RMP aims to generate feature codes that are unique to corner, edge and flat portions of an image. RCP, CSP and RCSP feature descriptors have been proposed for overcoming some of the limitations of the existing methods such as CP, LGC and it's variants.

In chapter 5, new feature descriptors inspired by the shape of various graphs have been presented for facial feature extraction. WGFD methods extracts features by encoding the

adjacent pixel relationship and the neighboring pixel relationship with the reference pixel in a local neighborhood. PGBP extracts features based on the vertices and edges of GPG(6,2) graph, whereas, LTrP (mTP and MTP) methods extract features by considering the triangles in both vertical and horizontal directions. Novel local texture based feature descriptors FFNND ($FFNND_1$ and $FFNND_2$), inspired by the structure of a feed forward neural network have been presented in chapter 6. The main objective of the proposed FFNND methods is to extract salient features in a local neighborhood with lesser fv length.

In chapters 3, 4 & 5, for CP, kTP, KTP, RMP, RCP, CSP, RCSP and WGFD methods, the experiments have been conducted with different weights to determine the optimal recognition accuracy. In chapter 6, the experiments have been conducted with different block sizes using both OVO-SVM and OVA-SVM classifiers for determining the optimal recognition accuracy. For all of the proposed methods mentioned in chapters 3, 4, 5 and 6, the experiments have been performed with respect to six, seven, eight and ten expressions for different FER datasets to validate the performance of the proposed methods. The experimental results demonstrated that the efficiency of the proposed methods when compared to the recent existing methods.

7.2 Future Scope

- **Different weights:** In general, the local based methods apply binary weights in the process of feature extraction. By using other weights such as fibonacci, prime, natural, squares and odd, the feature vector length can be greatly reduced. In this thesis, the concept of different weights has only been applied to FER problem. The possibility of using different weights for feature extraction in various image processing applications can be further explored.
- **FER combined with ageing problem:** Among the factors that make FER difficult, the most discriminating one is the age. Because of age related structural changes in the face, the expressions of older people appeared more difficult to encode, supporting the theory that wrinkles and folds in older faces actually resemble emotions. As a result, the state-of-the-art methods based on hand-crafted feature extraction may be

inadequate for the classification of FER performed by elderly people. Hence, FER combined with ageing is a problem that is worth studying in the future.

- **FER with super resolution images:** The Super Resolution (SR) task is used to create a higher resolution image from a low resolution image while attempting to fill in the lost pixels and avoiding the pixels becoming blurred. 'In the wild' FER datasets have images with small sizes and most of the CNN's are sensitive to input image size. In such cases, SR can be used for up-scaling the images and combined with deep networks for boosting up the performance.
- **Explainable AI:** Humans are the best judges for classifying and explaining any human emotion. But, in the case of AI, it displays the results of what it has learned without explaining such results. In the field of medical image analysis, AI can predict whether a person has pneumonia or not, by simply looking at an X-ray. However, it will not be trusted in the end because, it does not provide any explanation, which is critical, and instead suggests consulting with a doctor before announcing the final results. Whereas, Explainable AI will provide output and explain it's results in such cases, making it far more reliable than previous AI models. In the same manner, the concept of Explainable AI can be applied to FER for better explaining the output.

Author's Publications

Journals:

1. Mukku Nisanth Kartheek, Munaga V.N.K. Prasad and Raju Bhukya, "Chess Pattern with Different Weighting Schemes for Person Independent Facial Expression Recognition." *Multimedia Tools and Applications (Springer)*, 1-34, 2021.
   DOI: https://doi.org/10.1007/s11042-021-11270-8.

2. Mukku Nisanth Kartheek, Munaga V.N.K. Prasad and Raju Bhukya, "Radial Mesh Pattern: A Handcrafted Feature Descriptor for Facial Expression Recognition." *Journal of Ambient Intelligence and Humanized Computing (Springer)*, 1-13, 2021. DOI: https://doi.org/10.1007/s12652-021-03384-6.

3. Mukku Nisanth Kartheek, Munaga V.N.K. Prasad and Raju Bhukya, "Modified chess patterns: handcrafted feature descriptors for facial expression recognition." *Complex & Intelligent Systems (Springer)*, 7:3303-3322, 2021.
   DOI: https://doi.org/10.1007/s40747-021-00526-3.

4. Mukku Nisanth Kartheek, Munaga V.N.K Prasad and Raju Bhukya, "Local Triangular Patterns: Novel Handcrafted Feature Descriptors for Facial Expression Recognition." *International Journal of Biometrics (Inderscience)*, 2021. (In Press)

5. Mukku Nisanth Kartheek, Munaga V.N.K Prasad and Raju Bhukya, "Windmill Graph based Feature Descriptors for Facial Expression Recognition." *Optik*, 169053, 2022. DOI: https://doi.org/10.1016/j.ijleo.2022.169053.

6. Mukku Nisanth Kartheek, Munaga V.N.K Prasad and Raju Bhukya, "Feed Forward Neural Network Structure Inspired Handcrafted Feature Descriptors for Facial Expression Recognition". *Engineering Applications of Artificial Intelligence (Elseveir).* (To be communicated)

Conferences:

1. Mukku Nisanth Kartheek, Rapolu Madhuri, Munaga V.N.K Prasad and Raju Bhukya, "Knight Tour Patterns: Novel Handcrafted Feature Descriptors for Facial Expression Recognition" *In Proceedings of the 2021 19th International Conference on Computer Analysis of Images and Patterns*, pp. 210-219, 2021.

2. Mukku Nisanth Kartheek, Munaga V.N.K Prasad and Raju Bhukya, "Petersen Graph based Binary Pattern for Person Independent Facial Expression Recognition" *9th International Conference on Pattern Recognition and Machine Intelligence* ISI Kolkata, 2021. (Accepted)

Bibliography

[1] Lawrence O'Gorman. Comparing passwords, tokens, and biometrics for user authentication. *Proceedings of the IEEE*, 91(12):2021–2040, 2003.

[2] Anil K Jain, Patrick Flynn, and Arun A Ross. *Handbook of biometrics*. Springer Science & Business Media, 2007.

[3] Anil K Jain, Arun A Ross, and Karthik Nandakumar. *Introduction to biometrics*. Springer Science & Business Media, 2011.

[4] Salil Prabhakar, Sharath Pankanti, and Anil K Jain. Biometric recognition: Security and privacy concerns. *IEEE security & privacy*, 1(2):33–42, 2003.

[5] Anil K Jain, Arun Ross, and Salil Prabhakar. An introduction to biometric recognition. *IEEE Transactions on circuits and systems for video technology*, 14(1):4–20, 2004.

[6] Anil K Jain, Ruud Bolle, and Sharath Pankanti. *Biometrics: personal identification in networked society*, volume 479. Springer Science & Business Media, 2006.

[7] Ruud M Bolle, Jonathan H Connell, Sharath Pankanti, Nalini K Ratha, and Andrew W Senior. *Guide to biometrics*. Springer Science & Business Media, 2013.

[8] Susan R Fussell. The verbal communication of emotion: Introduction and overview. In *The verbal communication of emotions*, pages 9–24. Psychology Press, 2002.

[9] I Michael Revina and WR Sam Emmanuel. A survey on human face expression recognition techniques. *Journal of King Saud University-Computer and Information Sciences*, 1(5):1–9, 2018.

[10] Md Tauhid Bin Iqbal, M Abdullah-Al-Wadud, Byungyong Ryu, Farkhod Makhmudkhujaev, and Oksam Chae. Facial expression recognition with neighborhood-aware edge directional pattern (nedp). *IEEE Transactions on Affective Computing*, 11(1):125–137, 2018.

[11] Mukku Nisanth Kartheek, Munaga V N K Prasad, and Raju Bhukya. Local optimal oriented pattern for person independent facial expression recognition. In *Twelfth International Conference on Machine Vision (ICMV 2019)*, volume 11433, pages 114330R1–8. International Society for Optics and Photonics, 2020.

[12] Usman Saeed. Facial micro-expressions as a soft biometric for person recognition. *Pattern Recognition Letters*, 143:95–103, 2021.

[13] Abhay L Kashyap, Sergey Tulyakov, and Venu Govindaraju. Facial behavior as a soft biometric. In *2012 5th IAPR International Conference on Biometrics (ICB)*, pages 147–151. IEEE, 2012.

[14] Ying-Li Tian, Takeo Kanade, and Jeffrey F Cohn. Facial expression analysis. In *Handbook of face recognition*, pages 247–275. Springer, 2005.

[15] Farkhod Makhmudkhujaev, M Abdullah-Al-Wadud, Md Tauhid Bin Iqbal, Byungyong Ryu, and Oksam Chae. Facial expression recognition with local prominent directional pattern. *Signal Processing: Image Communication*, 74:1–12, 2019.

[16] Durga Ganga Rao Kola and Srinivas Kumar Samayamantula. A novel approach for facial expression recognition using local binary pattern with adaptive window. *Multimedia Tools and Applications*, 80(2):2243–2262, 2021.

[17] Murari Mandal, Monu Verma, Sonakshi Mathur, Santosh Kumar Vipparthi, Subrahmanyam Murala, and Deveerasetty Kranthi Kumar. Regional adaptive affinitive patterns (radap) with logical operators for facial expression recognition. *IET Image Processing*, 13(5):850–861, 2019.

[18] Ciprian Adrian Corneanu, Marc Oliu Simón, Jeffrey F Cohn, and Sergio Escalera Guerrero. Survey on rgb, 3d, thermal, and multimodal approaches for facial expression recognition: History, trends, and affect-related applications. *IEEE transactions on pattern analysis and machine intelligence*, 38(8):1548–1568, 2016.

[19] Peter Burkert, Felix Trier, Muhammad Zeshan Afzal, Andreas Dengel, and Marcus Liwicki. Dexpression: Deep convolutional neural network for expression recognition. *arXiv preprint arXiv:1509.05371*, 2015.

[20] Behzad H Mohammad Mahoor et al. Facial expression recognition using enhanced deep 3d convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 30–40, 2017.

[21] Gerard Pons and David Masip. Supervised committee of convolutional neural networks in automated facial expression analysis. *IEEE Transactions on Affective Computing*, 9(3):343–350, 2017.

[22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[23] Evangelos Sariyanidi, Hatice Gunes, and Andrea Cavallaro. Automatic analysis of facial affect: A survey of registration, representation, and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(6):1113–1133, 2014.

[24] Hai Hong, Hartmut Neven, and Christoph Von der Malsburg. Online facial expression recognition based on personalized galleries. In *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 354–359. IEEE, 1998.

[25] Irene Kotsia and Ioannis Pitas. Facial expression recognition in image sequences using geometric deformation features and support vector machines. *IEEE transactions on image processing*, 16(1):172–187, 2006.

[26] Maja Pantic and Ioannis Patras. Dynamics of facial expression: Recognition of facial actions and their temporal segments from face profile image sequences. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(2):433–449, 2006.

[27] Nicu Sebe, Michael S Lew, Yafei Sun, Ira Cohen, Theo Gevers, and Thomas S Huang. Authentic facial expression analysis. *Image and Vision Computing*, 25(12):1856–1863, 2007.

[28] Chih-Chin Lai and Chung-Hung Ko. Facial expression recognition based on two-stage features extraction. *Optik-International Journal for Light and Electron Optics*, 125(22):6678–6680, 2014.

[29] Songfan Yang and Bir Bhanu. Facial expression recognition using emotion avatar image. In *Face and Gesture 2011*, pages 866–871. IEEE, 2011.

[30] Min Hu, Yaqin Zheng, Chunjian Yang, Xiaohua Wang, Lei He, and Fuji Ren. Facial expression recognition using fusion features based on center-symmetric local octonary pattern. *IEEE Access*, 7:29882–29890, 2019.

[31] Adin Ramirez Rivera, Jorge Rojas Castillo, and Oksam Oksam Chae. Local directional number pattern for face analysis: Face and expression recognition. *IEEE transactions on image processing*, 22(5):1740–1752, 2012.

[32] Adín Ramírez Rivera, Jorge Rojas Castillo, and Oksam Chae. Local directional texture pattern image descriptor. *Pattern Recognition Letters*, 51:94–100, 2015.

[33] Byungyong Ryu, Adín Ramírez Rivera, Jaemyun Kim, and Oksam Chae. Local directional ternary pattern for facial expression recognition. *IEEE Transactions on Image Processing*, 26(12):6006–6018, 2017.

[34] Farkhod Makhmudkhujaev, Md Tauhid Bin Iqbal, Byungyong Ryu, and Oksam Chae. Local directional-structural pattern for person-independent facial expression recognition. *Turkish Journal of Electrical Engineering & Computer Sciences*, 27(1):516–531, 2019.

[35] Abuobayda MM Shabat and Jules-Raymond Tapamo. Angled local directional pattern for texture analysis with an application to facial expression recognition. *IET Computer Vision*, 12(5):603–608, 2018.

[36] Matthew Turk and Alex Pentland. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition*, pages 586–587, 1991.

[37] Peter N. Belhumeur, João P Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):711–720, 1997.

[38] SL Happy and Aurobinda Routray. Automatic facial expression recognition using features of salient facial patches. *IEEE transactions on Affective Computing*, 6(1):1–12, 2014.

[39] Anima Majumder, Laxmidhar Behera, and Venkatesh K Subramanian. Automatic facial expression recognition system using deep network-based data fusion. *IEEE transactions on cybernetics*, 48(1):103–114, 2016.

[40] Zhengyou Zhang, Michael Lyons, Michael Schuster, and Shigeru Akamatsu. Comparison between geometry-based and gabor-wavelets-based facial expression recognition using multi-layer perceptron. In *Proceedings Third IEEE International Conference on Automatic face and gesture recognition*, pages 454–459. IEEE, 1998.

[41] Michel François Valstar, Ioannis Patras, and Maja Pantic. Facial action unit detection using probabilistic actively learned support vector machines on tracked facial point data. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops*, pages 76–84. IEEE, 2005.

[42] Kamran Etemad and Rama Chellappa. Discriminant analysis for recognition of human face images. *Josa a*, 14(8):1724–1733, 1997.

[43] Turker Tuncer, Sengul Dogan, Moloud Abdar, and Pawel Plawiak. A novel facial image recognition method based on perceptual hash using quintet triple binary pattern. *Multimedia Tools and Applications*, pages 1–21, 2020.

[44] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[45] Heng-Da Cheng and XJ Shi. A simple and effective histogram equalization approach to image enhancement. *Digital signal processing*, 14(2):158–170, 2004.

[46] Caifeng Shan, Shaogang Gong, and Peter W McOwan. Facial expression recognition based on local binary patterns: A comprehensive study. *Image and vision Computing*, 27(6):803–816, 2009.

[47] Michael Lyons, Shigeru Akamatsu, Miyuki Kamachi, and Jiro Gyoba. Coding facial expressions with gabor wavelets. In *Proceedings Third IEEE international conference on automatic face and gesture recognition*, pages 200–205. IEEE, 1998.

[48] Niki Aifanti, Christos Papachristou, and Anastasios Delopoulos. The mug facial expression database. In *11th International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS 10*, pages 1–4. IEEE, 2010.

[49] Patrick Lucey, Jeffrey F Cohn, Takeo Kanade, Jason Saragih, Zara Ambadar, and Iain Matthews. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *2010 ieee computer society conference on computer vision and pattern recognition-workshops*, pages 94–101. IEEE, 2010.

[50] Guoying Zhao, Xiaohua Huang, Matti Taini, Stan Z Li, and Matti PietikäInen. Facial expression recognition from near-infrared videos. *Image and Vision Computing*, 29(9):607–619, 2011.

[51] Li-Fen Chen and Yu-Shiuan Yen. Taiwanese facial expression image database. *Brain Mapping Laboratory, Institute of Brain Science, National Yang-Ming University, Taipei, Taiwan*, 2007.

[52] Ellen Goeleven, Rudi De Raedt, Lemke Leyman, and Bruno Verschuere. The karolinska directed emotional faces: a validation study. *Cognition and emotion*, 22(6):1094–1118, 2008.

[53] Michal Olszanowski, Grzegorz Pochwatko, Krzysztof Kuklinski, Michal Scibor-Rylski, Peter Lewinski, and Rafal K Ohme. Warsaw set of emotional facial expression pictures: a validation study of facial display photographs. *Frontiers in psychology*, 5:1–8, 2015.

[54] Job Van Der Schalk, Skyler T Hawk, Agneta H Fischer, and Bertjan Doosje. Moving faces, looking places: validation of the amsterdam dynamic facial expression set (adfes). *Emotion*, 11(4):907–920, 2011.

[55] Shan Li and Weihong Deng. Reliable crowdsourcing and deep locality-preserving learning for unconstrained facial expression recognition. *IEEE Transactions on Image Processing*, 28(1):356–370, 2018.

[56] Deepali Aneja, Alex Colburn, Gary Faigin, Linda Shapiro, and Barbara Mones. Modeling stylized character expressions via deep learning. In *Asian conference on computer vision*, pages 136–153. Springer, 2016.

[57] Durga G Rao Kola and Srinivas K Samayamantula. Facial expression recognition using singular values and wavelet-based lgc-hd operator. *IET Biometrics*, 10(2):207–218, 2021.

[58] R Arya and ER Vimina. Local triangular coded pattern: A texture descriptor for image classification. *IETE Journal of Research*, pages 1–12, 2021.

[59] Vitomir Struc Ioannis A. Kakadiaris, Constantine Kotropoulos and Deepak Kumar Jain. Special issue on deep learning techniques applied to faces, february 2021.

[60] Saranya Rajan, Poongodi Chenniappan, Somasundaram Devaraj, and Nirmala Madian. Facial expression recognition techniques: a comprehensive survey. *IET Image Processing*, 13(7):1031–1040, 2019.

[61] Alex Pentland, Baback Moghaddam, Thad Starner, et al. View-based and modular eigenspaces for face recognition. 1994.

[62] Irfan A. Essa and Alex Paul Pentland. Coding, analysis, interpretation, and recognition of facial expressions. *IEEE transactions on pattern analysis and machine intelligence*, 19(7):757–763, 1997.

[63] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[64] Peng Zhao-Yi, Zhu Yan-Hui, and Zhou Yu. Real-time facial expression recognition based on adaptive canny operator edge detection. In *2010 Second International Conference on MultiMedia and Information Technology*, volume 2, pages 154–157. IEEE, 2010.

[65] Esmaeil Kheirkhah and Zahra Sadri Tabatabaie. A hybrid face detection approach in color images with complex background. *Indian Journal of Science and Technology*, 8(1):49–60, 2015.

[66] Zhao-yi Peng, Yu Zhou, and Ping Wang. Multi-pose face detection based on adaptive skin color and structure model. In *2009 International Conference on Computational Intelligence and Security*, volume 1, pages 325–329. IEEE, 2009.

[67] Madhusudhana Gargesha and Sethuraman Panchanathan. Face detection from color images by iterative thresholding on skin probability maps. In *2002 IEEE International Symposium on Circuits and Systems. Proceedings (Cat. No. 02CH37353)*, volume 5, pages V–V. IEEE, 2002.

[68] Kyung-Min Cho, Jeong-Hun Jang, and Ki-Sang Hong. Adaptive skin-color filter. *Pattern Recognition*, 34(5):1067–1073, 2001.

[69] SL Happy, Anjith George, and Aurobinda Routray. A real time facial expression classification system using local binary patterns. In *2012 4th International conference on intelligent human computer interaction (IHCI)*, pages 1–5. IEEE, 2012.

[70] Imran Hassan, Onaiza Maqbool, Qaisar Ahsan, and Usman Qayyum. Cascading neural network with adaboost for face detection. In *Int. Conf. Applied Sciences & Technology, Islamabad, Pakistan*, 2010.

[71] Mehul K Dabhi and Bhavna K Pancholi. Face detection system based on viola-jones algorithm. *International Journal of Science and Research (IJSR)*, 5(4):62–64, 2016.

[72] A Geetha, Vennila Ramalingam, S Palanivel, and B Palaniappan. Facial expression recognition–a real time approach. *Expert Systems with Applications*, 36(1):303–308, 2009.

[73] Aniruddha Dey. A contour based procedure for face detection and tracking from video. In *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, pages 483–488. IEEE, 2016.

[74] Olufisayo S Ekundayo and Serestina Viriri. Facial expression recognition: A review of trends and techniques. *IEEE Access*, 9:136944–136973, 2021.

[75] Feifei Zhang, Tianzhu Zhang, Qirong Mao, and Changsheng Xu. Geometry guided pose-invariant facial expression recognition. *IEEE Transactions on Image Processing*, 29:4445–4460, 2020.

[76] Rui Zhao, Tianshan Liu, Zixun Huang, Daniel Pak-Kong Lun, and Kenneth KM Lam. Geometry-aware facial expression recognition via attentive graph convolutional networks. *IEEE Transactions on Affective Computing*, 2021.

[77] Erfan Zangeneh and Aref Moradi. Facial expression recognition by using differential geometric features. *The imaging science journal*, 66(8):463–470, 2018.

[78] Ismail Oztel, Gozde Yolcu, Cemil Öz, Serap Kazan, and Filiz Bunyak. ifer: facial expression recognition using automatically selected geometric eye and eyebrow features. *Journal of Electronic Imaging*, 27(2):023003, 2018.

[79] Garima Sharma, Latika Singh, and Sumanlata Gautam. Automatic facial expression recognition using combined geometric features. *3D Research*, 10(2):1–9, 2019.

[80] Neha Jain, Shishir Kumar, and Amit Kumar. Effective approach for facial expression recognition using hybrid square-based diagonal pattern geometric model. *Multimedia Tools and Applications*, 78(20):29555–29571, 2019.

[81] Hajar Chouhayebi, Jamal Riffi, Mohamed Adnane Mahraz, Ali Yahyaouy, Hamid Tairi, and Nawal Alioua. Facial expression recognition based on geometric features. In *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*, pages 1–6. IEEE, 2020.

[82] Zoran Nenadic. Information discriminant analysis: Feature extraction with an information-theoretic objective. *IEEE transactions on pattern analysis and machine intelligence*, 29(8):1394–1407, 2007.

[83] Hui Zhou, Runsheng Wang, and Cheng Wang. A novel extended local-binary-pattern operator for texture analysis. *Information Sciences*, 178(22):4314–4325, 2008.

[84] Ying Tong, Rui Chen, and Yong Cheng. Facial expression recognition algorithm using lgc based on horizontal and diagonal prior principle. *Optik*, 125(16):4186–4189, 2014.

[85] Hsin-Wen Kung, Yi-Han Tu, and Chiou-Ting Hsu. Dual subspace nonnegative graph embedding for identity-independent expression recognition. *IEEE Transactions on Information Forensics and Security*, 10(3):626–639, 2015.

[86] Zhe Sun, Zheng-ping Hu, Meng Wang, and Shu-huan Zhao. Individual-free representation-based classification for facial expression recognition. *Signal, Image and Video Processing*, 11(4):597–604, 2017.

[87] Abubakar M Ashir and Alaa Eleyan. Facial expression recognition based on image pyramid and single-branch decision tree. *Signal, Image and Video Processing*, 11(6):1017–1024, 2017.

[88] Sadia Arshid, Ayyaz Hussain, Asim Munir, Anum Nawaz, and Sanneya Aziz. Multistage binary patterns for facial expression recognition in real world. *Cluster Computing*, 21(1):323–331, 2018.

[89] Monu Verma, Prafulla Sexena, Santosh Vipparthi, and Girdhari Singh. Quest: Quadriletral senary bit pattern for facial expression recognition. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1498–1503. IEEE, 2018.

[90] Swapna Agarwal, Bikash Santra, and Dipti Prasad Mukherjee. Anubhav: recognizing emotions through facial expression. *The visual computer*, 34(2):177–191, 2018.

[91] Jucheng Yang, Xiaojing Wang, Shujie Han, Jie Wang, Dong Sun Park, and Yuan Wang. Improved real-time facial expression recognition based on a novel balanced and symmetric local gradient coding. *Sensors*, 19(8):1899, 2019.

[92] M Kas, Y Ruichek, R Messoussi, et al. Multi level directional cross binary patterns: New handcrafted descriptor for svm-based texture classification. *Engineering Applications of Artificial Intelligence*, 94:103743, 2020.

[93] Hamid Sadeghi and Abolghasem-A Raie. Human vision inspired feature extraction for facial expression recognition. *Multimedia Tools and Applications*, 78(21):30335–30353, 2019.

[94] YB Ravi Kumar, CK Narayanappa, and P Dayananda. Weighted full binary tree-sliced binary pattern: An rgb-d image descriptor. *Heliyon*, 6(5):e03751, 2020.

[95] Subhadeep Koley, Hiranmoy Roy, and Debotosh Bhattacharjee. Gammadion binary pattern of shearlet coefficients (gbpsc): An illumination-invariant heterogeneous face descriptor. *Pattern Recognition Letters*, 145:30–36, 2021.

[96] Monu Verma, Prafulla Saxena, Santosh Kumar Vipparthi, and Girdhari Singh. Cross-centroid ripple pattern for facial expression recognition. *arXiv preprint arXiv:2201.05958*, 2022.

[97] Taskeed Jabid, Md Hasanul Kabir, and Oksam Chae. Robust facial expression recognition based on local directional pattern. *ETRI journal*, 32(5):784–794, 2010.

[98] Md Tauhid Bin Iqbal, Byungyong Ryu, Gihun Song, Jaemyun Kim, Farkhod Makhmudkhujaev, and Oksam Chae. Exploring positional ternary pattern (ptp) for conventional facial expression recognition from static images. *Korea Comput.Congress*, pages 853–855, 2016.

[99] Ying Tong and Rui Chen. Local dominant directional symmetrical coding patterns for facial expression recognition. *Computational intelligence and neuroscience*, 2019:1–13, 2019.

[100] V Uma Maheswari, G Varaprasad, and S Viswanadha Raju. Local directional maximum edge patterns for facial expression recognition. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–9, 2020.

[101] Yanli Ji, Yuhan Hu, Yang Yang, Fumin Shen, and Heng Tao Shen. Cross-domain facial expression recognition via an intra-category common feature and inter-category distinction feature fusion network. *Neurocomputing*, 333:231–239, 2019.

[102] Siyue Xie, Haifeng Hu, and Yongbo Wu. Deep multi-path convolutional neural network joint with salient region attention for facial expression recognition. *Pattern Recognition*, 92:177–191, 2019.

[103] Bing-Fei Wu and Chun-Hsien Lin. Adaptive feature mapping for customizing deep learning based facial expression recognition model. *IEEE access*, 6:12451–12461, 2018.

[104] Hang Zhao, Qing Liu, and Yun Yang. Transfer learning with ensemble of multiple feature representations. In *2018 IEEE 16th International Conference on Software Engineering Research, Management and Applications (SERA)*, pages 54–61. IEEE, 2018.

[105] Clement Feutry, Pablo Piantanida, Yoshua Bengio, and Pierre Duhamel. Learning anonymized representations with adversarial neural networks. *arXiv preprint arXiv:1802.09386*, 2018.

[106] Shervin Minaee and Amirali Abdolrashidi. Deep-emotion: Facial expression recognition using attentional convolutional network. *arXiv preprint arXiv:1902.01019*, 2019.

[107] Zhe Sun, Zheng-ping Hu, Meng Wang, and Shu-huan Zhao. Dictionary learning feature space via sparse representation classification for facial expression recognition. *Artificial Intelligence Review*, 51(1):1–18, 2019.

[108] Monu Verma, Santosh Kumar Vipparthi, and Girdhari Singh. Hinet: Hybrid inherited feature learning network for facial expression recognition. *IEEE Letters of the Computer Society*, 2(4):36–39, 2019.

[109] Wael Mohammad Alenazy and Abdullah Saleh Alqahtani. Gravitational search algorithm based optimized deep learning model with diverse set of features for facial expression recognition. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–16, 2020.

[110] Huadong Li and Hua Xu. Deep reinforcement learning for robust emotional classification in facial expression recognition. *Knowledge-Based Systems*, page 106172, 2020.

[111] Siyue Xie, Haifeng Hu, and Yizhen Chen. Facial expression recognition with two-branch disentangled generative adversarial network. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(6):2359–2371, 2020.

[112] Mohan Karnati, Ayan Seal, Anis Yazidi, and Ondrej Krejcar. Fer-net: Facial expression recognition using deep neural net. *Neural Computing and Applications*, 33:9125–9136, 2021.

[113] Sumeet Saurav, Prashant Gidde, Ravi Saini, and Sanjay Singh. Dual integrated convolutional neural network for real-time facial expression recognition in the wild. *The Visual Computer*, pages 1–14, 2021.

[114] A Hariprasad Reddy, Kamakshaiah Kolli, and Y Lakshmi Kiran. Deep cross feature adaptive network for facial emotion classification. *Signal, Image and Video Processing*, pages 1–8, 2021.

[115] Venkata Rami Reddy Chirra, Srinivasulu Reddy Uyyala, and Venkata Krishna Kishore Kolli. Virtual facial expression recognition using deep cnn with ensemble learning. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–19, 2021.

[116] Xinqi Fan, Mingjie Jiang, Ali Raza Shahid, and Hong Yan. Hierarchical scale convolutional neural network for facial expression recognition. *Cognitive Neurodynamics*, pages 1–12, 2022.

[117] Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma. Pcanet: A simple deep learning baseline for image classification. *IEEE transactions on image processing*, 24(12):5017–5032, 2015.

[118] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[119] Zhe Sun, Raymond Chiong, and Zheng-ping Hu. Self-adaptive feature learning based on a priori knowledge for facial expression recognition. *Knowledge-Based Systems*, 204:106124, 2020.

[120] Yan Yan, Zizhao Zhang, Si Chen, and Hanzi Wang. Low-resolution facial expression recognition: A filter learning perspective. *Signal Processing*, 169:107370, 2020.

[121] V Dharanya, Alex Noel Joseph Raj, and Varun P Gopi. Facial expression recognition through person-wise regeneration of expressions using auxiliary classifier generative adversarial network (ac-gan) based model. *Journal of Visual Communication and Image Representation*, 77:103110, 2021.

[122] Ali Muhamed Ali, Hanqi Zhuang, and Ali K Ibrahim. An approach for facial expression classification. *International Journal of Biometrics*, 9(2):96–112, 2017.

[123] Tehmina Kalsum, Zahid Mehmood, Farzana Kulsoom, Hassan Nazeer Chaudhry, Amjad Rehman Khan, Muhammad Rashid, and Tanzila Saba. Localization and classification of human facial emotions using local intensity order pattern and shape-based texture features. *Journal of Intelligent & Fuzzy Systems*, (Preprint):1–21, 2021.

[124] Tehmina Kalsum, Syed Muhammad Anwar, Muhammad Majid, Bilal Khan, and Sahibzada Muhammad Ali. Emotion recognition from facial expressions using hybrid feature descriptors. *IET Image Processing*, 12(6):1004–1012, 2018.

[125] Fengyuan Wang, Jianhua Lv, Guode Ying, Shenghui Chen, and Chi Zhang. Facial expression recognition from image based on hybrid features understanding. *Journal of Visual Communication and Image Representation*, 59:84–88, 2019.

[126] SL Happy and Aurobinda Routray. Robust facial expression classification using shape and appearance features. In *2015 eighth international conference on advances in pattern recognition (ICAPR)*, pages 1–5. IEEE, 2015.

[127] Faisal Farooq, Jalal Ahmed, and Lihong Zheng. Facial expression recognition using hybrid features and self-organizing maps. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 409–414. IEEE, 2017.

[128] Javad Abbasi Aghamaleki and Vahid Ashkani Chenarlogh. Multi-stream cnn for facial expression recognition in limited training data. *Multimedia Tools and Applications*, 78(16):22861–22882, 2019.

[129] Debashis Sen, Samyak Datta, and Raman Balasubramanian. Facial emotion classification using concatenated geometric and textural features. *Multimedia Tools and Applications*, 78(8):10287–10323, 2019.

[130] Ruiqi Li, Jing Tian, and Matthew Chin Heng Chua. Facial expression classification using salient pattern driven integrated geometric and textual features. *Multimedia tools and applications*, 78(20):28971–28983, 2019.

[131] Dami Jeong, Byung-Gyu Kim, and Suh-Yeon Dong. Deep joint spatiotemporal network (djstn) for efficient facial expression recognition. *Sensors*, 20(7):1936, 2020.

[132] Yan Wang, Ming Li, Xing Wan, Congxuan Zhang, and Yue Wang. Multiparameter space decision voting and fusion features for facial expression recognition. *Computational Intelligence and Neuroscience*, 2020, 2020.

[133] Ivan Gogić, Martina Manhart, Igor S Pandžić, and Jörgen Ahlberg. Fast facial expression recognition using local binary features and shallow neural networks. *The visual computer*, 36(1):97–112, 2020.

[134] P Shanthi and S Nickolas. An efficient automatic facial expression recognition using local neighborhood feature fusion. *Multimedia Tools and Applications*, 80(7):10187–10212, 2021.

[135] Chang Liu, Kaoru Hirota, Junjie Ma, Zhiyang Jia, and Yaping Dai. Facial expression recognition using hybrid features of pixel and geometry. *Ieee Access*, 9:18876–18889, 2021.

[136] Jun Liu, Hongxia Wang, and Yanjun Feng. An end-to-end deep model with discriminative facial features for facial expression recognition. *IEEE Access*, 9:12158–12166, 2021.

[137] Arun A Ross, Karthik Nandakumar, and Anil K Jain. *Handbook of multibiometrics*, volume 6. Springer Science & Business Media, 2006.

[138] Arun Ross and Anil Jain. Information fusion in biometrics. *Pattern recognition letters*, 24(13):2115–2125, 2003.

[139] Kyong I Chang, Kevin W Bowyer, and Patrick J Flynn. An evaluation of multimodal 2d+ 3d face biometrics. *IEEE transactions on pattern analysis and machine intelligence*, 27(4):619–624, 2005.

[140] Cees GM Snoek, Marcel Worring, and Arnold WM Smeulders. Early versus late fusion in semantic video analysis. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 399–402, 2005.

[141] Derek A Pisner and David M Schnyer. Support vector machine. In *Machine learning*, pages 101–121. Elsevier, 2020.

[142] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[143] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[144] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[145] Kuldeep Singh Yadav and Joyeeta Singha. Facial expression recognition using modified viola-john's algorithm and knn classifier. *Multimedia Tools and Applications*, 79(19):13089–13107, 2020.

[146] Daniel Canedo and António JR Neves. Facial expression recognition using computer vision: a systematic review. *Applied Sciences*, 9(21):4678, 2019.

[147] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2):131–163, 1997.

[148] Kevin P Murphy et al. Naive bayes classifiers. *University of British Columbia*, 18(60):1–8, 2006.

[149] Ira Cohen, Nicu Sebe, FG Gozman, Marcelo Cesar Cirelo, and Thomas S Huang. Learning bayesian network classifiers for facial expression recognition both labeled and unlabeled data. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages I–I. IEEE, 2003.

[150] Sean R Eddy. Hidden markov models. *Current opinion in structural biology*, 6(3):361–365, 1996.

[151] Mayur Rahul, Narendra Kohli, Rashi Agarwal, and Sanju Mishra. Facial expression recognition using geometric features and modified hidden markov model. *International Journal of Grid and Utility Computing*, 10(5):488–496, 2019.

[152] Jinglian Liang, Chao Xu, Zhiyong Feng, and Xirong Ma. Hidden markov model decision forest for dynamic facial expression recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 29(07):1556010, 2015.

[153] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[154] Fatima Zahra Salmam, Abdellah Madani, and Mohamed Kissi. Facial expression recognition using decision trees. In *2016 13th International Conference on Computer Graphics, Imaging and Visualization (CGiV)*, pages 125–130. IEEE, 2016.

[155] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[156] Xiaorong Pu, Ke Fan, Xiong Chen, Luping Ji, and Zhihu Zhou. Facial expression recognition from image sequences using twofold random forest classifier. *Neurocomputing*, 168:1173–1180, 2015.

[157] Mostafa K Abd El Meguid and Martin D Levine. Fully automated recognition of spontaneous facial expressions in videos using random forest classifiers. *IEEE Transactions on Affective Computing*, 5(2):141–154, 2014.

[158] John Wright, Allen Y Yang, Arvind Ganesh, S Shankar Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE transactions on pattern analysis and machine intelligence*, 31(2):210–227, 2008.

[159] Stefanos Zafeiriou and Maria Petrou. Sparse representations for facial expressions recognition via l 1 optimization. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, pages 32–39. IEEE, 2010.

[160] Mohammad H Mahoor, Mu Zhou, Kevin L Veon, S Mohammad Mavadati, and Jeffrey F Cohn. Facial action unit recognition with sparse representation. In *2011 IEEE International Conference on Automatic Face & Gesture Recognition (FG)*, pages 336–342. IEEE, 2011.

[161] Mohammad Reza Mohammadi, Emad Fatemizadeh, and Mohammad H Mahoor. Intensity estimation of spontaneous facial action units based on their sparsity properties. *IEEE transactions on cybernetics*, 46(3):817–826, 2015.

[162] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[163] Turker Tuncer, Sengul Dogan, and Volkan Ataman. A novel and accurate chess pattern for automated texture classification. *Physica A: Statistical Mechanics and its Applications*, 536:122584, 2019.

[164] P Chandra Sekhar Reddy, P Vara Prasad Rao, P Kiran Kumar Reddy, and M Sridhar. Motif shape primitives on fibonacci weighted neighborhood pattern for age classification. In *Soft Computing and Signal Processing*, pages 273–280. Springer, 2019.

[165] BS Shashikiran, K Shaila, and KR Venugopal. Minimal block knight's tour and edge with lsb pixel replacement based encrypted image steganography. *SN Computer Science*, 2(3):1–9, 2021.

[166] Ian Parberry. An efficient algorithm for the knight's tour problem. *Discrete Applied Mathematics*, 73(3):251–260, 1997.

[167] BD Parameshachari, HT Panduranga, et al. Secure transfer of images using pixel-level and bit-level permutation based on knight tour path scan pattern and henon map. In *Cognitive Informatics and Soft Computing*, pages 271–283. Springer, 2021.

[168] Subrahmanyam Murala and QM Jonathan Wu. Local mesh patterns versus local binary patterns: biomedical image indexing and retrieval. *IEEE journal of biomedical and health informatics*, 18(3):929–938, 2013.

[169] S Shen and S.S. Si. Facial expression recognition based on lgc in 5×5 neighborhood. *Intelligent Computing Applications*, 7:47–48, 2017.

[170] Joseph A Gallian. A dynamic survey of graph labeling. *Electronic Journal of combinatorics*, 1(DynamicSurveys):DS6, 2018.

[171] P Shiladhar, AM Naji, and ND Soner. Computation of leap zagreb indices of some windmill graphs. *International Journal of Mathematics And its Applications*, 55:7, 2018.

[172] Yan-Li Qin, Binzhou Xia, and Sanming Zhou. Canonical double covers of generalized petersen graphs, and double generalized petersen graphs. *Journal of Graph Theory*, 97(1):70–81, 2021.

[173] Turker Tuncer, Sengul Dogan, and Fatih Ertam. A novel neural network based image descriptor for texture classification. *Physica A: Statistical Mechanics and its Applications*, 526:120955, 2019.