UNIVERSITY OF KWAZULU-NATAL

COLLEGE OF AGRICULTURE, ENGINEERING AND SCIENCE

Using Facial Expression Recognition for Crowd Monitoring

By:

Ross Philip Holder

212504400

Supervisor:

Prof Jules-Raymond Tapamo

In fulfillment of the academic requirements of the degree of Master of Science in Engineering, School of Engineering, University of KwaZulu-Natal

July 2017

Preface

The research discussed in this dissertation was performed at the University of KwaZulu-Natal under the supervision of Prof. Jules-Raymond Tapamo. I hereby declare that all materials incorporated in this dissertation are my own original work except where acknowledgment is made by name or in form of reference. The work contained herein has not been submitted in part or whole for a degree at any other university.

Ross Philip Holder
July 2017

Declaration - Supervisor

As the candidate's Supervisor I agree to the submission of this dissertation.

Jules-Raymond Tapamo

Declaration - Plagiarism

I, Ross Philip Holder, declare that

1. The research reported in this dissertation, except where otherwise indicated, is my original research.
2. This dissertation has not been submitted for any degree or examination at any other university.
3. This dissertation does not contain other person's data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
4. This dissertation does not contain other person's writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:
    (a) Their words have been re-written but the general information attributed to them has been referenced

    (b) Where their exact words have been used, then their writing has been placed in italics and inside quotation marks, and referenced.
5. This dissertation does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the dissertation and in the References sections.

____

Ross Philip Holder

Declaration - Publications

Details of contribution to publications that form part and/or include research presented in this dissertation:

Publication 1: R. P. Holder and J. R. Tapamo, “Improved Gradient Local Ternary Patterns for Facial Expression Recognition”, *EURASIP Journal on Image and Video Processing*, (2017), **2017**:42, doi: 10.1186/s13640-017-0190-5.

Publication 2: R.P Holder and J. R Tapamo, “Using Facial Expression Recognition for Crowd Monitoring”, *Lecture Notes in Computer Science (LNCS): The 8th Pacific-Rim Symposium on Image and Video Technology (PSIVT)*, (Wuhan, Hubei, P. R. China, 2017), (Under Review).

Ross Philip Holder

Acknowledgments

I would like to thank the PRISM/CSIR for funding this research.

I would like to thank my supervisor, Prof Jules-Raymond Tapamo, for his support and guidance in this topic.

Ross Philip Holder
July 2017

Abstract

In recent years, Crowd Monitoring techniques have attracted emerging interest in the field of computer vision due to their ability to monitor groups of people in crowded areas, where conventional image processing methods would not suffice. Existing Crowd Monitoring techniques focus heavily on analyzing a crowd as a single entity, usually in terms of their density and movement pattern. While these techniques are well suited for the task of identifying dangerous and emergency situations, such as a large group of people exiting a building at once, they are very limited when it comes to identifying emotion within a crowd. By isolating different types of emotion within a crowd, we aim to predict the mood of a crowd even in scenes of non-panic.

In this work, we propose a novel Crowd Monitoring system based on estimating crowd emotion using Facial Expression Recognition (FER). In the past decade, both FER and activity recognition have been proposed for human emotion detection. However, facial expression is arguably more descriptive when identifying emotion and is less likely to be obscured in crowded environments compared to body posture. Given a crowd image, the popular Viola and Jones face detection algorithm is used to detect and extract unobscured faces from individuals in the crowd. A robust and efficient appearance based method of FER, such as Gradient Local Ternary Pattern (GLTP), is used together with a machine learning algorithm, Support Vector Machine (SVM), to extract and classify each facial expression as one of seven universally accepted emotions (joy, surprise, anger, fear, disgust, sadness or neutral emotion). Crowd emotion is estimated by isolating groups of similar emotion based on their relative size and weighting.

To validate the effectiveness of the proposed system, a series of cross-validation tests are performed using a novel Crowd Emotion dataset with known ground-truth emotions. The results show that the system presented is able to accurately and efficiently predict multiple classes of crowd emotion even in non-panic situations where movement and density information may be incomplete. In the future, this type of system can be used for many security applications; such as helping to alert authorities to potentially aggressive crowds of people in real-time.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

2DPCA	Two Dimensional Principal Component Analysis
ASM	Active Shape Model
BDBN	Boosted Deep Belief Network
CCA	Canonical Correlation Analysis
CK	Cohn-Kanade
CV	Cross-Validation
DBN	Deep Belief Network
DR	Dimensionality Reduction
FER	Facial Expression Recognition
GLTP	Gradient Local Ternary Pattern
GWT	Gabor Wavelet Transform
ICA	Independent Component Analysis
K-NN	K - Nearest Neighbours
LBP	Local Binary Pattern
LDA	Linear Discriminant Analysis
LDP	Local Directional Pattern
LOO	Leave-One-Out
LTP	Local Ternary Pattern
NN	Neural Network
PCA	Principal Component Analysis
PI	Person Independent
RBF	Radial Basis Function
SVM	Support Vector Machine

Chapter 1

Introduction

1.1 Background

Crowd Monitoring is a topic of emerging interest in the field of computer vision and was born largely from the desire to monitor the nature of groups of individuals in crowded areas, where conventional image processing methods would not suffice [1]. Areas where Crowd Monitoring systems are commonly deployed include airport terminals, sports stadiums, and other public facilities that attract large crowds of people.

With the increasing threat of terrorism to national security [2][3], agencies tasked with public safety have welcomed new developments and research into Crowd Monitoring techniques. Crowd Monitoring can be used to aid law enforcement in recognizing and identifying crowds that may cause public disorder. Examples include identifying disorderly crowds of sports fans that may have gathered after a football match, or a group of disgruntled protesters that have taken to the street.

With the advent of social media platforms, such as Twitter, small gatherings can often gather momentum very quickly, evolving into large crowds that can be difficult to control [4]; this necessitates the need for advances in Crowd Monitoring techniques. Techniques for Crowd Monitoring are formally introduced in Section

2.1.

Facial Expression Recognition (FER) is a technique used to extract and classify emotion from an individual's facial expression. It is widely accepted that there are seven universally recognizable emotions as first identified by Ekman [5], namely: joy, surprise, anger, fear, disgust, sadness and neutral emotion. In this work we use FER to extract and classify emotion from individuals in a crowded environment. The individual emotions can be combined to estimate the emotion of the crowd. FER is formally introduced in Section 2.2.

1.2 Problem definition

Due to the difficulty associated with extracting individuals from a crowd, most Crowd Monitoring techniques focus heavily on analyzing crowds as a single entity. Many different holistic based methods of Crowd Monitoring have been proposed in current literature, such as analyzing crowd movement patterns, flow and density. While these approaches are well suited for the task of identifying emergency situations, such as a large group of people exiting a building at once or a crowd gathering around a fight, they are very limited when it comes to identifying the nature or mood of a crowd outside of scenes of panic.

Although most Crowd Monitoring approaches focus on using holistic based methods, object-level based approaches to Crowd Monitoring have also been proposed. In most object-level based approaches, the dominant movement pattern of the crowd is formed based on an assessment of the movement patterns of individuals within and around the crowd. Individuals whose movement pattern oppose the dominant crowd flow can then be identified. While these approaches can be used to identify criminals, such as pickpockets, who intentionally move through crowds in opposing direction and at a different pace; they are unable to identify the mood of the crowd.

A system that is able to autonomously identify the mood of a crowd in real-time dynamic environments is required.

1.3 Motivation

There is potential for aggressive crowds, fuelled by their sense of superiority in numbers [6], to vandalize and loot property while endangering the lives of innocent bystanders. By identifying the mood of a crowd in real-time, the system can help to potentially identify aggressive and disorderly crowds so that necessary measures, such as additional policing units, can be deployed to prevent further aggression and violence. In areas where policing units are limited, the system allows officials to concentrate available units on crowds of interest; maximizing their resources and efficiency. The system aims to ultimately improve public safety and security.

The system uses emotion to represent the mood of the crowd. Crowd emotion can be estimated at object-level using Facial Expression Recognition.

1.4 Main goal and specific objectives

The main goal of this work is to produce a Crowd Monitoring system that is able to estimate the emotion of a crowd in real-time using Facial Expression Recognition (FER).

The specific objectives of this work are as follows:

1. To conduct a literature review on Crowd Monitoring techniques and Facial Expression Recognition (FER).
2. To propose improvements on a robust and efficient method of FER; enhancing recognition accuracy and efficiency.
3. To produce a Crowd Emotion dataset with ground-truth emotion labels by placing individuals expressing different prototypic emotions in the same scene.

4. To implement an algorithm to estimate the emotion of a crowd in real-time.

1.5 Contributions

This work contributes to the existing knowledge domain in the following ways:

- Improvements to an existing robust and efficient method of Facial Expression Recognition (FER) are proposed. The improved method is tested using well-known facial expression datasets and shown to further improve recognition accuracy and efficiency compared to other state of the art methods in current literature.
- A novel Crowd Emotion dataset is produced by combining separate images of individuals expressing different prototypic emotions in the same scene. Ground-truth emotions for each crowd image are provided. The dataset may be used in future work for purposes such as FER, facial recognition, people counting and face detection in crowded scenes.
- A novel method of Crowd Monitoring is proposed for safety and security applications using Facial Expression Recognition (FER). The proposed method is able to identify the mood of a crowd even in scenes of non-panic.

1.6 Dissertation Outline

In Chapter 2, a literature review on existing methods of Crowd Monitoring and Facial Expression Recognition (FER) is presented. This includes an investigation into holistic and object-level based methods of Crowd Monitoring as well as appearance-based, geometric-based, and other state of the art methods of FER in current literature. In Chapter 3, improvements to a robust and efficient method of FER, Gradient Local Ternary Pattern (GLTP), are presented. A machine learning algorithm for classification of emotion is also discussed. Chapter 4 details the creation of a novel

Crowd Emotion dataset which includes ground-truth emotion labels. In Chapter 5, methodology for estimating crowd emotion based on the individual emotions within the crowd is presented. This includes work on face detection algorithms and graph theory. A full discussion on experimental results, datasets and the metrics used for testing is presented in Chapter 6. This includes testing of FER with optimal parameter selection, face detection and crowd emotion estimation. A conclusion and ideas for future work are given Chapter 7.

Chapter 2

Literature Review

This chapter presents a literature review on existing methods of Crowd Monitoring and Facial Expression Recognition (FER). Section 2.1 investigates some popular approaches to Crowd Monitoring. Identification of possible applications for current methodology and their limitations are presented. Section 2.2 investigates common and state of the art methods of Facial Expression Recognition (FER); strengths and weaknesses of each method are identified. A summary of the findings is presented in Section 2.3.

2.1 Crowd Monitoring

Existing Crowd Monitoring techniques mainly focus on two different approaches: holistic approach; where the crowd is analyzed as a whole and object-level approach; where the crowd is analyzed as a collection of individuals. Techniques from both approaches have been summarized in surveys presented in [7] and [8]. In Sections 2.1.1 and 2.1.2, we investigate some of the most popular methods identified by these surveys as well as their limitations. More recently, methods for detecting emotion in crowds have also been proposed. In Section 2.1.3, we investigate some of these approaches as well as their limitations.

2.1.1 Holistic-based methods of Crowd Monitoring

Due to the difficulty associated with extracting individuals from a crowd because of obvious occlusions; Holistic-based approaches have formed a large part of the ongoing investigation into Crowd Monitoring techniques. Holistic-based methods are not concerned with local information, such as the movement of an individual in the crowd, but with global information, such as the overall flow of the crowd.

Davies et. al [9] proposed an approach to distinguish stationary crowds from moving crowds using discrete Fourier transforms combined with a linear area transform function. In terms of the Fourier transforms, fast moving crowds exhibit high frequencies while stationary crowds exhibit constant frequency levels. Motion is measured at pixel-neighbourhood level which can then be aggregated to obtain motion for larger areas and used to identify crowd velocity.

Similarly aimed approaches are investigated in [10], [11] and [12]. These approaches include the use of (1) block matching algorithms, (2) analysis of optical flow using Hidden Markov Models (HMMs) and (3) detection of crowd instabilities based on Lagrangian particle dynamics respectively. Although different approaches were used, the main goal remains very similar to that of Davies et. al [9]; that is to determine the overall flow (movement) of a crowd.

Methods to determine the density of crowds are also investigated. One such method is proposed by Wu et al. [13]. This method uses texture analysis and Support Vector Machines (SVM) to estimate the density of a crowd. It is argued by the authors that density estimation can be used to identify scenes of emergencies due to overcrowding.

The most popular holistic-based methods of Crowd Monitoring are aimed at identifying the overall flow and density of a crowd. These type of methods are well suited for the task of identifying dangerous or emergency situations in crowded areas. For example, detecting a large group of people rapidly exiting a building at once could indicate that there is a fire or possible explosion indoors. On the other hand, detecting an increasingly densely packed crowd of people in a specific area could be used to indicate a crowd gathering around a possible fight or injured person.

While existing holistic-based methods of Crowd Monitoring are well suited for the task of identifying dangerous or emergency situations, they do not provide specific details of what type of event has occurred; for example, is there a fire or an explosion ? These methods are also extremely limited when it comes to identifying emotion in crowds; for example, they cannot distinguish between a peaceful crowd and an aggressive crowd.

2.1.2 Object-level-based methods of Crowd Monitoring

Junior et al. [7] note in their survey that crowds are identified when the density of people is sufficiently large to disable individual identification. They also note that people can lose their individualities in a crowd instead adopting the behaviour of the crowd. These revelations have however not stopped researchers from investigating methods of Crowd Monitoring on an individual basis (object-level). Although object-level analysis of crowds is less popular than holistic-based methods, some novel object-level-based approaches to Crowd Monitoring are discussed below.

In past years, much work has been done into finding dominant movement patterns from groups of individuals. One such work is that of Cheriyadat and Radke [14]. The work centers around clustering sets of low-level motion features into trajectories. Similar work has been carried out by both Rabaud & Belongie [15] and Brostow & Cipolla [16]; however, Cheriyadat and Radke [14] proposed implementing additional clustering rules such as calculating dominant movements based on the longest common subsequences. Extracting dominant movement patterns from crowds allowed them to identify unusual movements made by individuals.

Similar to Cheriyadat and Radke [14], Zhang et al. [17] proposed using a clustering method based on trajectory distance using longest common subsequences. However, unlike Cheriyadat and Radke [14], their clustering scheme does not require tracking image feature points. Instead it uses a more robust long range motion estimation technique, particle video, to obtain long range motion trajectories. Zhang et al. [17] tested their approach on real video sequences; finding that it could successfully identify both dominant and anomalous motions in crowded scenes. They concluded that particle video is more reliable at finding particle trajectory compared to other approaches that use optical flow to find motion features.

Wang et al. [18] proposed a novel unsupervised learning framework to model activities in crowded scenes. A hierarchical Bayesian model is used to connect three elements: low-level visual features, simple atomic activities (modeled as distributions over low-level visual features), and multi-agent interactions (modeled as distributions over atomic activities). The model is learned in an unsupervised fashion. Wang et al. [18] states that this system is able summarize typical activities and interactions in a scene, segment a video sequence both temporally and spatially and detect abnormal activities without human input (unsupervised).

Similar to holistic-based methods; object-level-based methods of Crowd Monitoring are mainly aimed at analyzing crowd flow. However, there is a notable and important distinction between the two types of techniques. While holistic-based methods focus on overall crowd flow only, object-level-based methods are aimed at identifying individuals that move abnormally compared to that of the overall crowd. For example, object-level-based techniques of Crowd Monitoring could potentially be used to identify a pick-pocket who directly enters a crowd from the opposite direction of travel at a different pace.

Although object-level-based methods of Crowd Monitoring are useful at identifying suspicious individuals moving in a crowd, they are very limited when it comes to emotion detection. Similar to holistic-based methods, existing object-level-based methods of Crowd Monitoring cannot identify the emotion of a crowd.

2.1.3 Emotion detection in crowds

So far the methods we have investigated have not been aimed at emotion detection in crowds. In this section, we investigate methods for emotion detection in crowds. Majority of the methods are used for security applications; such as identifying violent crowds. However, systems that analyze crowd interest are also investigated here.

Emotion detection using Dynamic Probabilistic Models

Baig et al. [19] proposed use of a dynamic probabilistic clustering technique to model a crowd's response to different events. The first step of the procedure was to discretize the space under consideration into smaller more acceptable areas. The Instantaneous Topological Map (ITM) algorithm was employed to segment the observation space into zones of correlated trajectories. Bio-inspired autobiographical memory (AM) is used to model peoples reaction to events in their environment. A probabilistic graphical model is used to describe the relationship between interactions of event, cause and effect. Emotions are scaled into positive (pleasing) and negative (displeasing) using a 2-dimensional map of mental space of emotions. A simulation model to produce evacuation and panic situations was designed based on a Social Force Model (SFM) to test the proposed method. The authors report that their proposed method was capable of detecting emotions with a high accuracy.

While this approach represents a novel approach to emotion detection of crowds, it is limited by the fact that it is only able to classify emotions as positive or negative. This binary classification of emotion makes this method very limited for the purposes of emotion detection in crowds and, as stated by the author, is mainly suited for detecting panic and evacuation situations.

Emotion-based classifiers for abnormality detection

Rabiee et al. [20] proposed using a trained set of emotion-based classifiers to represent the behaviour of a crowd. By using attributing emotions of the crowd, more descriptive models can be created for crowd behaviour. First, a novel crowd dataset annotated with crowd emotion and crowd behaviour is created. Next, low-level visual features are extracted and used to represent mid-level emotion attributes provided by the ground-truth emotion annotations. Emotion based feature vectors can then be trained to represent different behaviour classes using a latent Support Vector Machine (SVM) [21] [22].

The authors note that although there are many crowd behaviour datasets available online, they tend to have only a small number of video sequences taken under controlled conditions with limited behaviour classes. A novel dataset is thus proposed, consisting of 31 video sequences in total with approximately 44,000 normal and abnormal video clips taken from 7 existing popular crowd behaviour datasets. The videos are first annotated with 5 different behaviour types (neutral, obstacles, panic, fight and congestion) then, because no emotion annotations are included, the authors ask a group of annotators to classify each scene in terms of 6 basic emotions (angry, happy, excited, scared, sad and neutral). The proposed dataset thus contains normal and abnormal crowd video sequences annotated with both crowd behaviour and crowd emotion.

A number of state of the art feature descriptors were proposed for extracting low-level visual features from the video clips; namely histogram of oriented gradients (HOG), histogram of optical flow (HOF), motion boundary histogram (MBH) and dense trajectories. Each method was tested separately on the dataset using the leave-one-out testing procedure. The authors found that dense trajectories achieved the highest results on test with an average recognition accuracy of 43.9%. They note that their results show that higher accuracies are achieved by combining emotion and behaviour classifiers compared to using each approach on its own. They also note that their dataset can be used in future work of the same nature.

While this method serves as a novel approach to detecting emotion in crowds, it still has some significant limitations. Firstly, rather than classifying unseen crowds by an emotion, this method instead attempts to classify them into a behaviour class based on the ground-truth emotions mapped to behaviour classes in the training set. So although this method inherently uses emotion to classify crowd behaviour, it does not explicitly label the emotion of crowd test data. Secondly, although the video sequences range from neutral to apex crowd emotions, it is likely that only the apex scenes that will contain enough low-level discriminative features to distinguish between emotions. For example, in order to distinguish between a sad and angry crowd, video sequences near the apex scenes will have to be used as the scenes before this point will likely look very similar; this is because of the high holistic-based level at which the images are examined (i.e. the whole scene). This means that a violent crowd, for example, will only be identified after it has already reached a high level of aggression. Finally, the behaviour of a crowd may be expressed differently depending on the environment, for example, an indoor protest vs. an outdoor protest; this means that recognition accuracy will suffer unless the trained data includes video sequences from many different environments.

Emotion analysis of social media

Ngo et al. [23] proposed using emotion analysis of social media for emergency management at mass gatherings. The authors argue that having information about the type of crowd at an event can highly assist with the response in case of an emergency. With the emergence of social media, more information about how people think and feel can be captured. The authors thus proposed using emotion analysis of social media to identify crowd types at an event. Experimental tests using historical data were conducted to validate the effectiveness of the proposed system. The authors note that although some existing work, such as that in [24], has been undertaken into identifying crowd types using social media analysis; emotion analysis of social media has not yet been used to represent crowd types.

The proposed system framework consists of 3 main stages. Firstly, contextual data is extracted from social media, such as Twitter, using geolocation data and hashtags. This allows social media to be filtered based on a specific event. In the second stage, emotion analysis of the social media is performed using a simplified representation of the bag-of-words approach [25]. In this approach, the frequency appearance of keywords is used to model the media into high and low levels of four distinct categories of emotion; namely anger, fear, happiness and sadness. Finally, the emotions and their corresponding levels obtained in the second stage are used to classify the crowd type into one of five groups based on a crowd type mapping model. The five groups of crowd types [26] identified by the authors are (1) ambulatory, limited movement and spectator (2) expressive/cohesive and participatory (3) aggressive, demonstrator and violent (4) escaping and dense/suffocating and (5) rushing/looting.

To test the proposed framework, a high profile sporting event was chosen where a known stampede occurred. Social media pertaining to the event was collected using geolocation data as well as the hashtag of the event from Twitter. The tweets were analyzed at a rate of one tweet per fifteen minutes from the period during and after the event. The authors note that an emotion was successfully extracted from 84.67% of the total tweets. The authors also note that their results show that there was a significant increase in the proportion of tweets labeled with the fear emotion during the stampede. The fear emotion is closely correlated to the escaping and dense/suffocating crowd type i.e. a stampede. The authors thus state that their proposed framework was able to successfully identify the crowd type during an emergency situation at a mass gathering. The authors also make the point that their system is not affected by operating conditions, such as illumination, like other computer vision-based methods and does not require deploying any hardware or software-based equipment at the venue.

With that said, there are still numerous limitations with the proposed framework. Firstly, the system relies on filtering social media using geolocation information and/or hashtags. This means that users are required to provide this information as part of their social media messages. It is not guaranteed that users will include this information and some may not even use social media to begin with, which will in turn reduce the accuracy of the system. Secondly, the proposed system can only monitor crowds at specific, largely mainstream, events. This system could not be deployed in relatively unknown public areas. Finally, although the system can potentially analyze social media in real-time, messages relating to emergencies are likely to come after the emergency has already occurred. While this can improve the response, early detection is limited and emergencies cannot be prevented from occurring in the first place.

Detecting the interest-level of crowds

So far, the methods discussed in this section have been aimed towards improving public safety and security by identifying emergency situations and negative crowd behaviours using emotion. However, an existing crowd emotion monitoring system, known as SWEET [27], has been proposed to enhance public experiences at conferences and other events. Rather than providing enhanced measures of safety and security to the public, the SWEET system instead uses crowd emotion to identify the level of interest experienced by groups of people attending different presentations or events. Members of the public can then be directed to most interesting presentation or event that is taking place.

The system works by taking photos of individuals attending different events through a smartphone application. The facial expressions of the individuals in the photo are then analyzed to determine their emotion. This, combined with location information and surrounding sounds, is used to determine the interest level of the people attending the presentation or event. This information can then be used to help other attendees make the best use of their time by attending only the most interesting events.

The biggest limitation with this system is that, in its current form at least, it cannot be used for security and safety applications. The fact that the system requires user submitted images via a smartphone application and is heavily dependent on accurate location information to aggregate the images, makes the system unfeasible for autonomous real-time security applications. However, determining crowd emotion from facial expressions for security and safety applications is a novel idea that is investigated in more detail throughout this body of work.

Existing approaches to crowd emotion detection do not offer a significant breakthrough in identifying the distinct emotions present in a crowd. We thus propose a novel Crowd Monitoring technique that is able to identify the moment a crowd starts exhibiting signs of aggression. This will be achieved by estimating the emotion of the crowd in real time based on the individual emotions present in the crowd. Zhan et al. [1] notes that conventional computer vision techniques are not appropriate when analyzing crowded scenes. However, in Section 2.2, we investigate conventional methods of emotion detection with the aim of applying these methods to form a novel emotion-based Crowd Monitoring system.

2.2 Facial Expression Recognition (FER)

Over the past decade, automated human emotion detection has been a topic of significant interest in the field of computer vision. Two of the most common approaches to emotion detection are human behaviour analysis [28][29] and Facial Expression Recognition (FER) [30][31][32]. In behaviour analysis, an individual's emotion is determined by analyzing their stance and movement patterns, while in FER, emotion is determined from the individual's facial expression. In this work, we use FER to determine the emotion of an individual in the crowd. FER is arguably more descriptive than behaviour analysis [30] and an individual's face is also less likely to be obscured in crowded areas. Due to the wide variety of individuals that could be found in a crowd; an accurate, efficient and robust method of FER is required for the purposes of Crowd Monitoring. Much work has been carried out over the past decade into creating and improving high accuracy, robust methods of FER. Some novel FER techniques are investigated from Sections 2.2.2 to 2.2.6.

2.2.1 Overview

FER consists mainly of three important steps [30]; (1) face detection and pre-processing, (2) facial feature extraction, and finally (3) expression classification. In the first step, faces are identified and extracted from the background. Different regions of the face can then be extracted such as eyebrows, eyes, nose and mouth. The Viola and Jones face detection algorithm [33][34] is widely used due to its efficiency, robustness and accuracy at identifying faces in uncontrolled backgrounds. Other methods include the use of Active Shape Models (ASM) [35][36][37] to identify facial points and edges.

In the second step, suitable features that are able to describe the emotion of the face are extracted. The features are grouped into two main categories; appearance and geometric. In appearance-based methods, an image filter is applied to the whole face or specific facial regions to extract changes in texture due to specific emotions; such as wrinkles and bulges. Common appearance-based feature extraction methods include the use of Local Binary Patterns (LBP) [36][38][39], Local Directional Patterns (LDP) [39][40], Local Ternary Patterns (LTP) [41], Gradient Local Ternary Patterns (GLTP) [42] and Gabor Wavelet Transforms (GWT) [30][39][43]. Geometric methods focus on extracting features that measure the distance between certain points on the face such as the distance between corners of the eye and mouth. The shape of various facial components due to changing emotions can also be extracted. Geometric-based feature extraction is often considered more difficult to implement than appearance-based methods due to the variability in the size and shape of features across emotions [30].

Because of the large amount of data that can be extracted from a face, a significant portion constitutes redundant information. Large feature sets can greatly reduce program efficiency; especially when training a classifier. Dimensionality Reduction (DR) techniques are often used to reduce the size of the feature set by removing the redundant data; greatly improving efficiency without reducing accuracy. Common DR techniques include the use of Principle Component Analysis (PCA) [30][34][40][44], Independent Component Analysis (ICA) [30], Linear Discriminant Analysis (LDA) [30] and AdaBoost [40][45].

The final step of FER is to create a classifier based on the features extracted in step two. The extracted features are fed into a machine learning algorithm that attempts to classify them into distinct classes of emotion based on the similarities between feature data. Once the classifier has been trained, it is used to assign input features to a particular class of emotion, for example [5]; joy, surprise, anger, fear, disgust, sadness or neutral emotion. Common supervised classifiers include Support Vector Machines (SVM) [30][34][36][37][40][42][46], K-Nearest Neighbours (K-NN) [39][47] and Neural Networks (NN) [30][37][39][48].

2.2.2 Appearance-based methods of FER

Spatial domain versus transform domain

Suja and Tripathi [39] investigated the use of spatial domain methods versus transform domain methods for feature extraction on cropped and whole faces. The authors tested LBP [38] and LDP [40] (discussed further on) as spatial domain methods and Dual Tree-Complex Wavelet Transform (DT-CWT) [49] and Gabor Wavelet Transform (GWT) [43] as transform domain methods. Feature extraction was performed on whole faces (segmented from background) and cropped faces (eye, eyebrows and mouth are segmented from the face). Classification was performed using NN [48] and K-NN [47].

DT-CWT is an improvement to Discrete Wavelet Transform (DWT). DT-CWT uses two real DWTs; the first of which gives the real part of the transform and the second of which gives the imaginary part thus finding the complex transform of the image. The two transforms use two different sets of filters that allow for perfect reconstruction and shift invariance. DT-CWT produces sub-bands of the image made up of complex coefficients. The highest 500 high pass sub-band coefficients and all low frequency coefficients (which contain the most significant information) are used to form the feature vectors.

Gabor Wavelet Transform (GWT) [43] is a special form of short time Fourier transform. A Gabor Wavelet is a function that minimizers its standard deviation in the frequency and time domains thus having the lowest theoretically possible uncertainty bound. Suja and Tripathi convoluted each input image with generated Gabor kernels for five scales and eight orientations to obtain Gabor wavelets. The magnitude of all coefficients for every scale and orientation was found together with the mean and standard deviation of 40 Gabor wavelets. Feature vectors were formed by combining the mean and standard deviation of the coefficients.

The features were trained separately using a NN and K-NN classifier. The NN consisted of an input layer, a hidden layer with 20 neurons and an output layer. The number of neurons in the input and output layers are dependent on the feature set size and number of emotions. The NN was trained using the trainscg algorithm. The activations of the input neurons, based on the feature vectors, are passed on to other neurons until finally the output neuron activates; which determines the emotion. In the K-NN classifier, the Euclidean distance (shortest distance) between each test sample and the trained data was calculated. The test data was classified as the class where the distance was found to be a minimum.

Suja and Tripathi tested their method using the JAFFE [50] and CK [51][52] datasets.

For the JAFFE dataset, 70 images were used as the training set and another 70 images as the test set for 7-classes of emotion. For the CK dataset, 540 images were used as the training set and 180 images as the test set for 6-classes of emotion. According to the authors results, transform domain methods outperformed spatial domain methods when used with NN and K-NN classifiers. For majority of the results, using cropped faces (facial regions such as the eyes and mouth) yielded more accurate results than using the whole face. Although the authors found that transform domain methods outperformed spatial domain methods when used together with NN and K-NN; spatial domain methods such as Local Binary Pattern (LBP) [38] and Local Directional Pattern (LDP) [40] have been proven more effective when used with SVM [36][40] classifiers. In our work, we use three commonly referenced testing procedures as in [34] and [53] (discussed in Chapter 6). The validity of the results reported by Suja and Tripathi are questionable as they did not use a comparable testing procedure to others in literature; which could explain why the results reported for the JAFFE dataset are so poor (see Table 2.1).

Local Binary Pattern (LBP)

Wang et al. [36] proposed a novel FER method based on geometric alignment and LBP features. The system uses an ASM [35] to align faces, LBP [38] to extract features and a SVM [30][46] classifier to predict emotions. ASM uses a statistical approach for shape modeling based on a Point Distribution Model (PDM) which can label face edges, eyes and mouth regions accurately as well as align the face horizontally.

LBP was applied to the face to extract histograms that represent the texture of the face. LBP forms labels for each image pixel (represented by a grey-level intensity value) by thresholding a circular neighbourhood of P sampling points on a radius R of each pixel with the centre value. LBP makes use of a 2-level discrimination encoding scheme, where pixels larger than the centre value are assigned a value of

+1 and pixels smaller than the centre value are assigned a value of 0. The result can be considered a binary number with 2^P binary patterns being produced. Uniform patterns (binary patterns that contain at most two bitwise transitions) are used to reduce the size of the feature vector. Each uniform pattern is given a separate label while non-uniform patterns are given a single label. Histograms are calculated from the labels in different regions of the face and concatenated to form the LBP feature vector.

A SVM classifier was used to classify the LBP feature vectors. SVM is, by nature, a linear classifier that maximizes the margin between decision hyperplane and features from the training set. An optimal hyperplane is formed that minimizes the classification error of test samples. The proposed method was tested on the JAFFE [50] dataset using a 5-fold cross-validation testing scheme. A recognition accuracy of 86.1% was achieved for six basic emotions while 83.7% accuracy was achieved for 7 basic emotions (includes neutral expression).

The authors note that while their method achieved higher accuracies than other methods, such as using Gabor wavelets [30][43]; more advanced feature extraction methods are capable of achieving higher accuracies [40]. While LBP, which uses grey-level intensity values to encode the texture of an image, is computationally efficient, it has been shown to perform poorly under the presence of non-monotonic illumination variation and random noise as even a small change in grey-level values can easily change the LBP code [54].

Local Directional Pattern (LDP)

Jabid et al. [40] proposed a novel appearance-based feature descriptor; Local Directional Pattern (LDP). LDP employs a different texture coding scheme to that of LBP, where directional edge response values are used instead of grey-level intensity values. The use of dimensionality reduction techniques such as PCA [30][44] and AdaBoost [45] were also investigated. SVM [30][46] was used as a classifier and tests were performed on the JAFFE [50] and CK [51][52] datasets.

Local Directional Pattern (LDP) [40] was developed as an improvement to LBP [38] which is sensitive to non-monotonic illumination variation and random noise. LDP computes the edge response values in 8 directions at each pixel; encoding them into 8 bit binary numbers based on the relative strength of the edge responses. As edge responses are less sensitive to noise and illumination than intensity values; LDP is more robust than LBP. The image is split up into multiple regions and histograms are formed from the codes obtained for each region. An LDP feature vector is formed by concatenating the histograms from each area.

To reduce the dimension of the feature vector, PCA is used to transform the existing features into a new vector of reduced features. Firstly, eigenvectors are computed from the covariance data matrix. Each feature is approximated by a linear combination of the topmost few eigenvectors. The weight coefficients represent the eigenspace, and the eigenvalues that are close to zero can be discarded. The eigenvectors associated with the top eigenvalues defines the reduced subspace and the original feature vector can be projected onto this subspace to reduce its size. AdaBoost is another method of dimensionality reduction that can be used to select a subset of the existing feature set. AdaBoost works by learning a small number of weak classifiers whose performance is slightly better than random guessing and boosts them iteratively into a strong accurate classifier.

Both dimensionality reduction techniques were applied and tested separately using a Radial Basis Function (RBF) kernel SVM. PCA was shown to be the more effective dimensionality reduction technique achieving a higher recognition accuracy with a smaller feature set. Testing was performed on the JAFFE and CK datasets using a 7-fold cross-validation testing scheme. The authors report that their proposed method achieved an average recognition accuracy of 96.4% (6-class CK), 93.4% (7-class CK), 90.1% (6-class JAFFE) and 85.4% (7-class JAFFE).

Although the findings show that LDP outperforms LBP across both datasets, LDP still tends to produce inconsistent patterns in uniform and near-uniform regions due to its two-level discrimination encoding scheme and is heavily dependent on the number of prominent edge directions parameter [55].

Local Ternary Pattern (LTP)

Although LDP represents a considerable improvement to LBP, both techniques produce inconsistencies in uniform and near-uniform regions because of their use of a two-level discrimination encoding scheme. This is especially prevalent in FER as majority of facial regions are considered to be near-uniform. To solve this limitation, Local Ternary Pattern (LTP) [41] was proposed for facial features selection.

In contrast to LBP and LDP, LTP implements a three-level discrimination encoding scheme to ensure consistency in uniform and near-uniform facial regions. Grey-level intensity values falling in between $+t$ and $-t$ of the center value are quantized to 0, while those falling above $+t$ and below $-t$ are quantized to $+1$ and -1 respectively. The binary code in LBP is thus extended to a three-valued ternary code in LTP. One consequence with using a three-level encoding scheme is that the number of possible patterns increases from the usual 2^8 patterns in LBP to 3^8 patterns in LTP. To solve this issue, the ternary code is split into positive and negative parts and each is treated as individual binary patterns. Histograms from both the positive and negative codes are concatenated to form the feature vector.

In [42], LTP was implemented and tested on the CK dataset using a SVM. Results were compared with LBP and LDP methods using a 10-fold cross-validation testing scheme under the same experimental setup. The results showed that LTP performed similarly to LDP, with LTP achieving a recognition accuracy of 93.6% (6-class) & 88.9% (7-class) compared to 93.7% (6-class) & 88.4% (7-class) for LDP. Both methods outperformed LBP. While LTP increases the robustness of LBP and LDP in uniform and near-uniform regions, it still makes use of grey-level-based intensity values. This means that LTP, like LBP, is still adversely affected by non-monotonic illumination variation and random noise.

Gradient Local Ternary Pattern (GLTP)

Ahmed and Hossain [42] proposed a novel appearance-based feature descriptor for FER using gradient-based ternary texture patterns. Gradient Local Ternary Pattern (GLTP) was proposed to encode the local texture of an image by computing the gradient magnitudes of local neighbourhoods in the image and quantizing the values into three discrimination levels. The location and frequency occurrence information (histograms) of the resulting micropatterns were used as facial feature descriptors. A SVM was used for classification and the proposed method was tested on the CK dataset.

The authors note that while LDP [40] and LTP [41] offer improvements to LBP [36]; neither method provides an optimal solution in solving the limitations of LBP. LDP uses more robust edge response values compared to intensity values but implements only a two-level discrimination encoding scheme. On the other hand, LTP implements a robust three-level discrimination encoding scheme but still makes use of sensitive intensity values. GLTP was proposed to address these limitations by combining the advantages of Sobel-LBP [56] and LTP [41]. GLTP thus makes use of gradient magnitude values which are robust against illumination variation and random noise and implements a three-level ternary encoding scheme to improve robustness in uniform and near-uniform regions.

In GLTP, the gradient magnitudes of each pixel position in the image are calculated using the Sobel operator [56]. In uniform or near-uniform regions the gradient magnitudes will be very similar whereas in high-textured regions the magnitudes of edge pixels will be relatively higher. To differentiate between smooth and high-textured areas a 3×3 threshold of $\pm t$ is applied around the centre gradient value (G_c) of each neighbourhood. Neighbour values falling in between $G_c + t$ and $G_c - t$ are quantized to 0, while those below $G_c - t$ and above $G_c + t$ are quantized to -1

and +1 respectively. Because the number of possible GLTP patterns is large (3^8); each code is split into positive and negative parts. Each facial image is split into multiple regions and the GLTP histogram for each region is calculated by combining the histograms of the positive and negative codes for the corresponding region. Finally, the GLTP histograms from each region are concatenated to form the GLTP feature vector. The feature vector is passed to a RBF kernel SVM for training and classification.

The proposed method is tested using a 10-fold cross-validation testing scheme on the CK dataset. The threshold value (t) was empirically selected and three different region sizes of 3×3, 5×5 and 7×6 were tested. The authors note that greater accuracies are achieved with larger region sizes but at greater computational cost. An accuracy of 97.2% was achieved for 6-classes of emotion and 91.7% for 7-classes of emotion using 7×6 facial regions. The findings show that GLTP outperforms LBP, LDP and LTP in terms of recognition accuracy on the CK dataset while still maintaining a high level of computational efficiency. The superiority of GLTP lies in the utilization of robust gradient magnitude values with a three-level ternary encoding scheme; ensuring consistent texture patterns even under the presence of non-monotonic illumination variation and random noise.

A common limitation with all appearance-based methods of FER, including GLTP, is that the number of features extracted from images tends to be very large. This is to be expected as most appearance-based features selection methods use histograms to represent encoded patterns within the image and each histogram alone can hold up to 256 features. This, combined with the fact that most methods further segment the image into multiple regions, can result in an unnecessarily large amount of information. Take GLTP for example, if we segment the image into 7×6 regions and find a positive and negative histogram, each consisting of 256 features, for each region; the resulting feature vector can have up to: 7×6 regions $\times$ 2 histograms $\times$ 256 features $= 21504$ features. Unfortunately, most of these features are likely to constitute redundant information as not every region of the image is guaranteed to contain the same amount of discriminative data. Having such a large feature set will likely reduce classification efficiency and can also reduce recognition accuracy. One of the most common ways to solve this problem, is to use dimensionality reduction techniques to reduce the size of the feature vector so that only the most discriminative data remains. In the sections above, we have already discussed some methods of dimensionality reduction, such as PCA and Adaboost.

2.2.3 Geometric-based methods of FER

Bezier Curves

Bao and Ma [37] proposed a feature extraction method based on Bezier curves for FER. The authors note that appearance-based feature extraction methods that analyze the whole face are accurate but classification can take a long time due to a large amount of useless calculation. Bezier curve feature extraction was proposed to solve this issue. The proposed method was tested on the JAFFE [50] dataset using both SVM [46] and NN [48] classifiers.

Firstly, an ASM [35] was used to align the face and locate facial components such as eyes, eyebrows, nose and mouth. Next a Bezier curve [57] was used to characterize key components of the face. Key components that produce the most discriminative expression features were identified to be the eyebrows, eyes and mouth. A Bezier curve can be used to accurately represent the shape of each smooth key component under different expressions using four control points. The distance and angle of the end-points of the Bezier curve relative to a neutral expression template was used as the feature vector for each expression.

Testing was performed on the JAFFE dataset for 6 classes of emotion using both a SVM and NN classifier. 191 randomly selected images from the dataset were used as the training set and the remaining 23 images as the test set. Bao and Ma reported a recognition accuracy of 95.65% when using a SVM and 91.3% when using a NN. The findings once again demonstrate that a SVM classifier outperforms a NN classifier for the task of FER. Unfortunately, the authors did not make use of a common, comparable testing procedure such as those in [34] and [53]. However, if we were to disregard any variations due to the different testing procedure used; we note that the proposed method achieved a high level of recognition accuracy when used together with a SVM classifier.

Although geometric-based methods of FER improves the computational efficiency of classification by having a reduced feature vector compared to those of appearance-based methods; a high recognition accuracy can still be achieved as shown in [37]. However, as we have already noted, geometric methods can be difficult to implement due to the large variability in the size and shape of facial features across different emotions and subjects [30]. For example, the shape of one subject's eye can vary greatly from another subject's eye even if they are expressing the same emotion. It is for this reason, that geometric-based methods of FER can be considered less robust than their appearance-based counterparts.

2.2.4 Multiple feature sets for FER

DCT, SIFT and Gabor wavelets

Shaukat et al. [34] proposed using multiple feature sets for FER. Firstly, faces were extracted using the popular Viola and Jones [33] face detection algorithm and resized to ensure uniform size. Next, a combination of three different feature extraction methods were used to extract discriminative features from different regions of the face. The authors identified Discrete Cosine Transform (DCT) [58], Scale Invariant Features Transform (SIFT) [59], and Gabor wavelets [60] as three methods that are effective at recognizing emotion from faces. PCA [44] was used to reduce the dimensionality of the feature sets. Finally, the feature sets were concatenated and classified using a SVM [46]. The proposed method was extensively tested on the JAFFE [50] dataset.

DCT is a frequency domain transform that is used to describe an image as coefficients of different frequencies of cosines. DCT is comprised of three frequency components (low, middle and high) each of which gives some detail of the image. Each face was divided into an upper and lower region. The upper region consisted of eyes and eyebrows (components that can move and rotate) while the lower region consisted of mouth and lips (components that can be reshaped). DCT was applied to both the upper and lower region of the face and a DCT feature vector was formed by combining the two feature sets.

SIFT was used to detect matching features within images. To avoid extracting features affected by noise and other obscurities, as well as reducing the computational cost, SIFT was only applied to four subsections of the face; the left-eye, right-eye, nose and mouth. These subsections were found using the Viola and Jones algorithm. To further improve efficiency, the features from each subsection were combined and PCA was applied to reduce dimensionality; forming a new reduced SIFT feature vector.

The whole facial image was used to construct Gabor wavelets. Important fiducial points were extracted from the face using filters constructed from Gabor kernels for five different scales and eight orientations. The average of forty images convolved with multiple spatial resolutions, having multiple orientation sets of Gabor filters, was used as the feature set. PCA was applied to reduce dimensionality forming a reduced Gabor wavelet feature vector.

The DCT, SIFT and Gabor feature vectors were concatenated and passed to a RBF-kernel SVM for training and classification. The proposed method was tested on the JAFFE dataset using 10-fold cross-validation (CV), leave-one-out (LOO) and person independent (PI) testing procedures introduced in [53]. The authors report an average recognition accuracy of 99.04% (CV), 100% (LOO) and 91.03% (PI) for 7-classes of emotion. The authors note that their results are the highest achieved on the JAFFE dataset in current literature.

Although the results show that the proposed method achieves a high level of recognition accuracy, it is unclear what contribution each individual feature set made. Forming three different feature sets and applying dimensionality reduction separately is likely to have an adverse affect on computational efficiency. Thus if one of the methods contributes very little to improving overall recognition accuracy, it may prove beneficial to remove it and instead increase computational efficiency. Another limitation of using multiple feature sets is that multiple forms of pre-processing are required; one for each independent method. Again, this not only increases complexity, but increases the computational cost of the system.

Canonical Correlation Analysis (CCA) for transform domain features fusion

El-Shazly et al. [61] proposed a novel FER method using Canonical Correlation Analysis (CCA) for transform domain features fusion and classification. Firstly, transform domain methods were used to extract features from a face. Secondly, two-dimensional PCA (2DPCA) [62] was used to reduce the size of each feature set. Finally, the reduced features were fused together and classified using CCA [63].

The authors proposed using a combination of common transform domain techniques to extract features. Fast Fourier Transform (FFT), Discrete Cosine Transform (DCT), and Discrete Wavelet Transform (DWT) were selected. By using multiple transforms, more features can be obtained which improves classification accuracy however increases the amount of resources required for classification. To enhance efficiency, 2DPCA, a more efficient form of PCA, was applied to each set of transform features to remove redundant data and decrease the computational cost of classification.

CCA was applied to fuse together the feature sets from each transform and for classification. The basic idea behind CCA is to analyze the relationship between two sets of variables; finding two new basis vectors such that the correlation between the projections of the variables onto the vectors is maximized. The merit of CCA lies in the fact that it is able to take multiple modalities data into consideration which improves classification accuracy. CCA transforms both training and testing feature sets into new, highly correlated, sets. When testing, the training set that gives the highest correlation is the target set.

The proposed method was tested on the JAFFE [50] dataset. Two images for each of the 7 expressions of each subject were used as the training set and the remaining 73 images were used as the test set. The methods DWT-CCA, DCT-CCA, FFT-CCA and the combination of all three, (DWT, DCT, FFT)-CCA, achieved 94.52%, 93.15%, 95.89% and 97.26% accuracy respectively. The results clearly show that the highest recognition accuracy was achieved by combining all three feature extraction techniques.

The main difference between this proposed method and that in [34], besides the different feature extraction techniques used, is that in this method, CCA was used for features fusion and classification compared to a traditional classifier, such as SVM. One of the main limitations with CCA is that it is strictly a linear-based classifier and assumes that all data is linearly related [64]. Any outliers can thus cause severe problems when used together with CCA. In these tests, CCA was reported to have achieved high recognition accuracies however, SVM classifiers can be considered more robust as they offer kernels for data that is not linearly separable [46]. The validity of the reported results also comes into question as the authors did not use a common comparable testing procedure as in [34] and [53].

Although using multiple feature sets for FER can increase recognition accuracy; the extra amount of data included in the combined feature vector can make the classification process slow. In both [34] and [61], dimensionality reduction (DR) was proposed to solve this issue by reducing the feature vector size. However, DR must be performed on each individual feature set before they are combined. This process can greatly reduce efficiency, especially if it is to be performed on a crowd of faces.

In [34], the need to perform three different types of pre-processing steps per face will also harm system efficiency if it is to be used in a real-time environment. Finally, in certain situations, having an additional feature set may only result in a slight increase in recognition accuracy; an improvement that is not worth the resulting increase in computational cost.

2.2.5 Deep Learning for FER

Deep Belief Networks (DBN)

Lv et al. [65] proposed implementing FER via Deep Learning. They note that different parts of the face contain different amounts of discriminative information and that the weighting varies across faces. To combat this, they proposed recognizing emotion from only the facial components that are active during expression using Face Parsing (FP) [66] detectors. Parsing components removes redundant information and ensures that images don't need alignment. FP detectors were trained via a Deep Belief Network (DBN) [67] and tuned by logistic regression. The face, nose, eyes and mouth are detected hierarchically. The concentrated features of the detected components are classified using a deep architecture pre-trained with Stacked Autoencoder (SAE) [68]. The proposed method was tested using both the JAFFE [50] and CK+ [51][52] datasets.

DBNs are probabilistic generative models that consist of several Restricted Boltzmann Machine (RBM) [69] layers where each layer captures strong high-order correlations between activities of hidden features in the layer below. The authors use DBN to establish strong correlations between images and shapes of each component by estimating label maps directly from the detected image patches.

SAE is a deep network consisting of an autoencoding neural network in each layer. SAE models are trained layerwise by learning an autoencoder that reconstructs the previous layer. After a stack of encoders have been trained, an output layer is added to the top of the stack. The expression classifier is a three layer hidden Neural Network pre-trained with SAEs.

Testing was carried out using a seven-fold cross-validation scheme for seven classes of emotion. The proposed method achieved an average recognition accuracy of 90.47% and 91.11% on the JAFFE and CK+ databases respectively. Among others, the authors compare their findings to the work of Liu et al. [70] who proposed using a Boosted Deep Belief Network (BDBN) for feature selection, learning and classification in a unified loopy framework. The authors note that although Liu et al. achieved a higher accuracy of 96.7% on the CK+ database; their testing was based on only six classes of emotion (as opposed to seven in Lv et al.s testing) and is far more computationally intensive.

By far the greatest and most widely debated limitation with Deep Learning methods is that of computational efficiency. Methods that employ Deep Learning techniques are widely considered to be less efficient than more traditional techniques, as noted in [65], due to their inherently multi-layered approach to feature extraction and classification. The level of complexity of Deep Learning methods, such as that in [65], is also high; due to the wide range of parameters that must be configured for multiple network layers. If the Deep Learning network is not configured correctly, recognition accuracy will greatly decrease. The results reported in [65] and [70] also do not show a considerable increase in recognition accuracy compared to much more computationally efficient methods of FER; such as that in [42].

2.2.6 Dynamic-based FER

So far this review has covered methods of FER for static images. However more recently, work has been undertaken into dynamic FER. Different to static FER, dynamic FER aims to determine emotion from an image sequence rather than a single static image. Dynamic FER offers improvements to existing methods as facial expressions in an image sequence contains both spatial (appearance) and temporal (expression evolution) information.

Atlas construction and Sparse Representation

Guo et al. [71] proposed a method of dynamic FER using Atlas construction and Sparse Representation. The proposed method consists mainly of two stages; the Atlas construction stage and the recognition stage. In the Atlas construction stage, a diffeomorphic [72] growth model was estimated for each image sequence to capture facial expression characteristics. This is motivated by the fact that facial expression can be described by diffeomorphic motions of muscles beneath the face [73]. Atlases are also capable of suppressing variations due to inter-subject facial shapes. Each image sequence was made sure to begin with the neutral expression and end with the apex of the expression. To reflect the overall evolution of each expression, longitudinal Atlases were constructed using groupwise registration [74] and Sparse representation [75]; which is capable of capturing subtle expression information. In the recognition stage, each sample image sequence was registered to the constructed Atlas of each expression using both appearance information and temporal evolution information.

The proposed method was tested using numerous dynamic facial expression datasets including the CK+ dataset [51][52]. Here, we only consider the CK+ dataset for comparison to other techniques. 325 sequences from 118 subjects performing 7 facial expressions were used for Atlas construction and testing was performed using the person independent testing procedure. The authors report an average recognition accuracy of 97.2%; higher than that of other dynamic methods in current literature.

The main strength of dynamic FER lies in the fact that it uses both spatial and temporal information for more accurate expression recognition. However, facial expression image sequences are not always guaranteed to be available. In [71], the same testing procedure was repeated without temporal information (i.e. only spatial information was used) and it was found that the average recognition accuracy decreased by 4.8% from 97.2% to 92.4%. This shows that when temporal information is lost, the accuracy of dynamic FER decreases significantly. Another limitation of dynamic FER is that of computational efficiency. It is fairly obvious that the computational efficiency of dynamic FER is less than that of static FER because, to achieve maximum accuracy, multiple image sequences need to be analyzed for each subject under test; which takes time. In our body of work it is also unfeasible to expect that a facial expression sequence be available for every subject in the crowd and that all sequences be analyzed in real-time. The authors report that their method took an average time of 1.6 seconds in the recognition stage for each query sequence using a modern PC (4-core, 2.5GHz processor and 6 GB RAM); further demonstrating that the proposed method is not suitable for a real-time environment.

2.3 Chapter Summary

In Section 2.1, we investigated existing methods of Crowd Monitoring. We started by looking at some novel holistic-based [9][10][11][12][13] and object-level-based [14][15][16][17][18] methods in current literature. We found that both approaches were unable to detect the mood of a crowd in scenes of non-panic. It was noted that some form of emotion analysis of the crowd would be needed to distinguish between peaceful and aggressive crowds. We thus investigated some novel methods aimed at analyzing emotion in crowds [19][20][23][27]. We found that these methods did not represent a significant breakthrough in terms of real-time identification of multiple classes of emotion in changing environments. We thus proposed using conventional methods of emotion detection to identify multiple classes of emotion in crowds in real-time.

We compared the use of human behaviour analysis and Facial Expression Recognition (FER) for emotion detection in crowds and concluded that FER was the more appropriate technique as it is arguably more descriptive than behaviour analysis and faces are less likely to be obscured in crowds. In Section 2.2, we investigated some methods of FER, the results of which are summarized in Table 2.1. Firstly, we investigated the use of appearance-based methods [39][36][40][41][42] for FER and found that they offer a high level of efficiency, robustness and accuracy. They do however, tend to contain a large amount of useless information which can slow down the classification process. Secondly, we investigated the use of a geometric-based method [37] for FER. It was found to provide a good level of accuracy but a low level of robustness due to the variability in the size and shape of facial components across different subjects. Thirdly, we investigated the use of multiple feature sets [34][61] for FER. We noted that the added complexity of requiring multiple pre-processing steps and reduced system efficiency may not be worth the improvement in recognition accuracy. Next, we investigated the use of Deep Learning [65][70] for FER. We found that while there was no significant improvement in recognition accuracy, system complexity was high and efficiency was very poor. Finally, we investigated a novel dynamic-based [71] method of FER. We found that recognition accuracy decreased significantly when temporal information was not available and also noted that analyzing an image sequence for each individual within the crowd was unfeasible for real-world applications.

In this work, we want to extract emotion in real-time from a diverse crowd of individuals in a wide range of environments. We thus require a robust, efficient and accurate method of FER. Based on the review conducted in Section 2.2, we identify the appearance-based method, Gradient Local Ternary Pattern (GLTP) [42], that meets all of the above requirements. GLTP is computationally efficient, robust against changing environments and has been shown to achieve a high level of recognition accuracy. However, as GLTP is an appearance-based method, the computed feature vector tends to be very large which can reduce the efficiency of classification. We thus propose using dimensionality reduction (DR) to reduce the size of the feature vector and improve classification efficiency. In [40], PCA was shown to outperform Adaboost as a DR technique. We compare the results in [42], where LDP was implemented without DR to those in [40], where LDP was implemented with DR (see Table 2.1). The results show that DR not only improves efficiency but can also improve recognition accuracy. For classification, results have shown that, for the task of FER, the robust and efficient Support Vector Machine (SVM) [36][37][40] classifier with a Radial Basis Function (RBF) kernel outperforms other classifiers such as Neural Networks (NN) [37][39] and K-Nearest Neighbours (KNN) [39].

Table 2.1: Comparison of recognition rates (%) for some methods of FER on JAFFE and CK datasets

	Proposed Methodology	Datasets			
		JAFFE [50]		CK [51][52]	
		6-class	7-class	6-class	7-class
A-b	DT-CWT + K-NN [39]	-	65[1]	99[2]	-
	DT-CWT + NN [39]	-	88.6[1]	99.9[2]	-
	GWT + K-NN [39]	-	54.3[1]	98.6[2]	-
	GWT + NN [39]	-	87.9[1]	98.3[2]	-
	LBP + K-NN [39]	-	58.6[1]	97.1[2]	-
	LBP + NN [39]	-	40.7[1]	81.1[2]	-
	LBP + SVM [36]	86.1[3]	83.7[3]	-	-
	LBP + SVM [42]	-	-	90.1[4]	83.3[4]
	LDP + K-NN [39]	-	50[1]	85.4[2]	-
	LDP + NN [39]	-	42.1[1]	76.5[2]	-
	LDP + SVM [42]	-	-	93.7[4]	88.4[4]
	LDP + DR + SVM [40]	90.1[5]	85.4[5]	96.4[5]	93.4[5]
	LTP + SVM [42]	-	-	93.6[4]	88.9[4]
	GLTP + SVM [42]	-	-	97.2[4]	91.7[4]
G-b	Bezier curves + NN [37]	91.3[6]	-	-	-
	Bezier curves + SVM [37]	95.7[6]	-	-	-
M f s	DCT + SIFT + Gabor wavelets [34]	-	99[4] 100[7] 91[8]	-	-
	(DWT, DCT, FFT) - CCA [61]	-	97.3[9]	-	-
D L	DBN [65]	-	90.5[5]	-	91.1[5]
	BDBN [70]	-	-	96.7[10]	-
Dyn	Atlas constr. & Sparse Repres. [71]	-	-	-	97.2[8] 92.4[8,11]

A-b = Appearance-based G-b = Geometric-based M f s = Multiple feature sets

D L = Deep Learning Dyn = Dynamic-based

[1] train-70, test-70

[2] train-540, test-180

[3] 5-fold cross-validation

[4] 10-fold cross-validation

[5] 7-fold cross-validation

[6] train-191, test-23

[7] leave-one-out

[8] person independent

[9] train-140, test-73

[10] 8-fold cross-validation

[11] without temporal info.

Chapter 3

Improved Gradient Local Ternary Pattern

3.1 Introduction

This chapter presents improvements to Gradient Local Ternary Pattern (GLTP). In Section 3.2, we discuss the robust and efficient method of feature extraction, GLTP, and explore potential improvements to enhance the method's accuracy and efficiency. The relevant pre-processing procedures for both GLTP & Improved GLTP are presented in Section 3.3. A method for classification is presented in Section 3.4.

3.2 Feature Extraction

Gradient Local Ternary Pattern (GLTP) is a local appearance-based facial texture descriptor proposed by Ahmed and Hossain [42]. GLTP is used to encode the local texture of a facial expression by calculating the gradient magnitudes of local neighbourhoods within the image and quantizing the values into three different discrimination levels. The resulting local patterns are used as facial feature descriptors.

GLTP aims to address the limitations of common appearance-based features like LBP [36][38] and LDP [40] by combining the advantages of the Sobel-LBP [56] and LTP [41] operators. GLTP uses more robust gradient magnitude values as opposed to grey levels with a three-level encoding scheme to discriminate between smooth and high textured facial regions. This ensures the generation of consistent texture patterns even under the presence of illumination variation and random noise.

Horizontal and vertical approximations of the derivatives of the source image $f(x, y)$ are obtained by applying the Sobel-Feldman [56][76] operator. A Sobel operator convolves the source image with a horizontal and vertical mask to obtain the horizontal (G_x) and vertical (G_y) approximations of the derivatives respectively (see Figure 3.1(a, b, c)). The gradient magnitude ($G_{x,y}$) (see Figure 3.1(d)) for each pixel can then be found by using the Euclidean norm as:

$$G_{x,y} = \sqrt{G_x^2 + G_y^2} \tag{3.1}$$

To differentiate between smooth and high textured facial regions, a threshold of $\pm t$ is applied around a centre gradient value (G_c) of 3×3 pixel neighbourhoods throughout the gradient magnitude image. Neighbour gradient values falling in between $G_c + t$ and $G_c - t$ are quantized to 0, while those below $G_c - t$ and above $G_c + t$ are quantized to -1 and $+1$ respectively. In other words, we have

$$S_{GLTP}(G_c, G_i) = \begin{cases} -1, & G_i < G_c - t \\ 0, & G_c - t \leq G_i \leq G_c + t \\ +1, & G_i > G_c + t \end{cases} \tag{3.2}$$

where G_c is the centre gradient value of a 3×3 neighbourhood, and G_i and S_{GLTP} are the gradient magnitude and quantized value of the surrounding neighbours respectively (see Figure 3.1(e, f)).

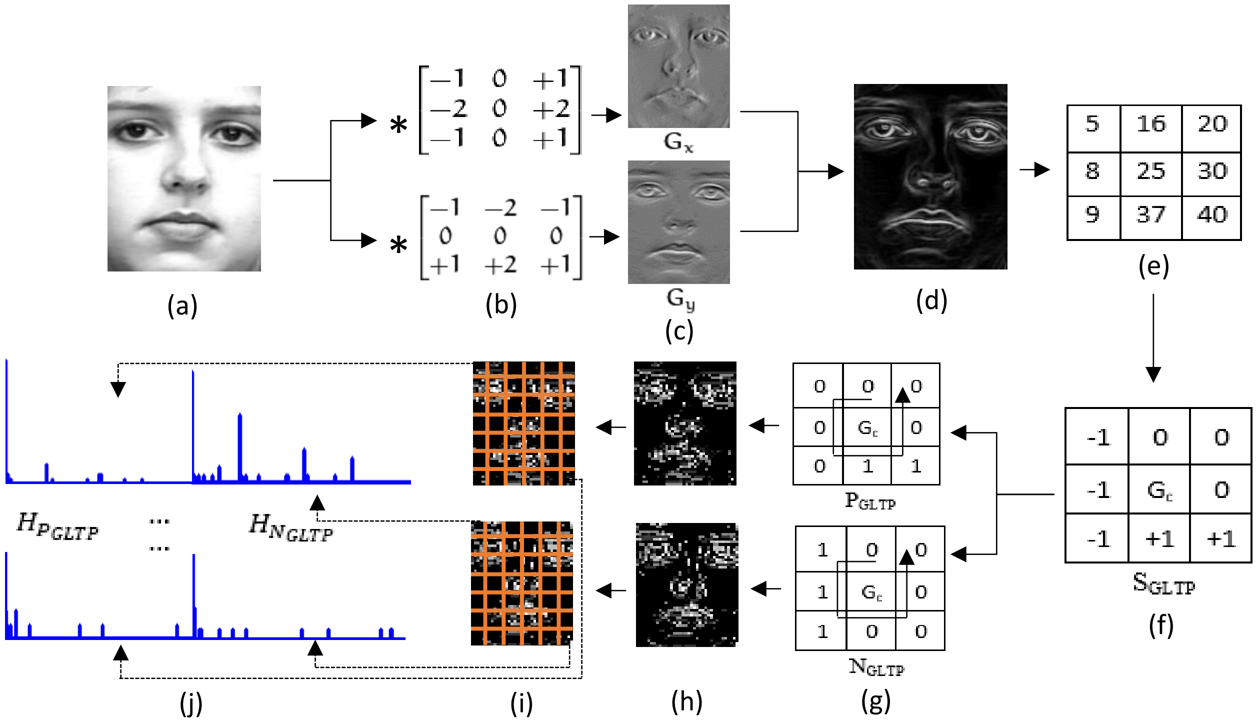


Figure 3.1: The GLTP process (a) Source image (b) Horizontal & vertical Sobel masks (c) Horizontal and vertical gradient images from Sobel operator (d) Gradient magnitude image (e) Sample 3×3 neighbourhood of gradient magnitudes (f) GLTP codes for $t = 10$ (g) Splitting GLTP code into positive and negative code (h) Positive and negative decimal coded image (i) Segmenting positive and negative images into 7×6 regions (j) Concatenate positive and negative GLTP histogram from each region to form feature vector

The resulting 8 S_{GLTP} values for each 3×3 neighbourhood can be concatenated to form a GLTP code. However, using a three-level discrimination coding scheme results in a much higher number of possible patterns (3^8) when compared to that of LBP (2^8) [36] which would result in a high dimensional feature vector. To reduce the dimensionality, each GLTP code is split into its positive and negative parts and treated as individual codes (see Figure 3.1(g)) as outlined by Tan and Triggs [41]. The formula for converting each binary GLTP code to positive (P_{GLTP}) and negative (N_{GLTP}) decimal codes are given in equations 3.3 and 3.4.

$$P_{GLTP} = \sum_{i=0}^{7} S_P(S_{GLTP}(i)) \times 2^i \tag{3.3}$$

$$S_P(v) = \begin{cases} 1, & \text{if } v > 0 \\ 0, & \text{otherwise} \end{cases}$$

$$N_{GLTP} = \sum_{i=0}^{7} S_N(S_{GLTP}(i)) \times 2^i \tag{3.4}$$

$$S_N(v) = \begin{cases} 1, & \text{if } v < 0 \\ 0, & \text{otherwise} \end{cases}$$

After computing the positive (P_{GLTP}) and negative (N_{GLTP}) GLTP decimal coded image representations (see Figure 3.1(h)), each image is divided into $m \times n$ regions (see Figure 3.1(i)). A positive ($H_{P_{GLTP}}$) and negative ($H_{N_{GLTP}}$) GLTP histogram is computed for each region using the equations:

$$H_{P_{GLTP}}(\tau) = \sum_{r=1}^{\frac{M}{m}} \sum_{c=1}^{\frac{N}{n}} f(P_{GLTP}(r,c), \tau) \tag{3.5}$$

$$H_{N_{GLTP}}(\tau) = \sum_{r=1}^{\frac{M}{m}} \sum_{c=1}^{\frac{N}{n}} f(N_{GLTP}(r,c), \tau) \tag{3.6}$$

$$f(\alpha, \tau) = \begin{cases} 1, & \text{if } \alpha = \tau \\ 0, & \text{otherwise} \end{cases}$$

where M and N are the width and height of the GLTP coded image, r and c represents row and column and τ is the GLTP code value (usually, 0-255) for which you are finding the frequency occurrence. By computing a GLTP histogram for each region; the location information of the GLTP micro-patterns are combined with their occurrence frequencies thus improving recognition accuracy.

Finally, the positive and negative GLTP histogram for each region are concatenated together to form the feature vector (see Figure 3.1(j)) represented as:

$$H_{P_{GLTP}}(1,1)H_{N_{GLTP}}(1,1)\ldots\ldots H_{P_{GLTP}}(m,n)H_{N_{GLTP}}(m,n) \tag{3.7}$$

The pseudo-code for implementing GLTP is presented in Algorithm 3.1.

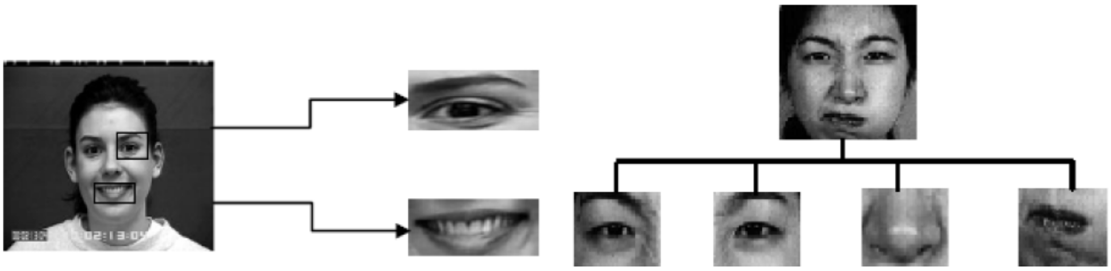
Although GLTP [42] has been shown to perform well for the task of FER, outperforming other appearance-based methods, we believe that recognition accuracy and efficiency can still be further improved. In Chapter 2, we noted that in some work [39][77], it was found that performing feature extraction on specific regions of the face yielded more accurate results than performing feature extraction on the whole face. We thus discuss the use of facial component extraction for feature selection in Section 3.2.1. In Section 3.2.2, we discuss the use of a more accurate gradient operator, the Scharr operator, for computing the gradient magnitude image. The most prominent problem with appearance-based methods is the large amount of redundant data that they compute. We thus discuss the use of dimensionality reduction in Section 3.2.3. The proposed improvements are integrated together to form Improved GLTP.

Algorithm 3.1 Gradient Local Ternary Pattern

```
Require: Pre-processed source image
Ensure: Vector of GLTP histograms

Initialize variables
    $t \leftarrow$ set the threshold

Compute horizontal ($G_x$) and vertical ($G_y$) derivative approximations of source image using Sobel operator

Compute the gradient magnitude for each pixel of the source image
    $G_{x,y} = \sqrt{G_x^2 + G_y^2}$

Apply threshold of $\pm t$ around centre gradient value ($G_c$) in a $3 \times 3$ neighbourhood to determine $S_{\mathbf{GLTP}}$ codes for the gradient magnitude image
    for each gradient magnitude value ($G_i$) around $G_c$ do
        if $G_i > G_c + t$ then $S_{\mathrm{GLTP}}(i) \leftarrow +1$
        else if $G_i < G_c - t$ then $S_{\mathrm{GLTP}}(i) \leftarrow -1$
        else$S_{\mathrm{GLTP}}(i) \leftarrow 0$
    repeat for each $3 \times 3$ neighbourhood

Compute positive ($P_{\mathbf{GLTP}}$) and negative ($N_{\mathbf{GLTP}}$) GLTP coded image representations from $S_{\mathbf{GLTP}}$ values
    for each $S_{\mathrm{GLTP}}(i)$ value in a $3 \times 3$ neighbourhood do
        if $S_{\mathrm{GLTP}}(i) > 0$ then $S_P(i) \leftarrow 1$ & $S_N(i) \leftarrow 0$
        else if $S_{\mathrm{GLTP}}(i) < 0$ then $S_P(i) \leftarrow 0$ & $S_N(i) \leftarrow 1$
        else$S_P(i) \leftarrow 0$ & $S_N(i) \leftarrow 0$
    $P_{GLTP} = \sum_{i=0}^{7} S_P(i) \times 2^i$
    $N_{GLTP} = \sum_{i=0}^{7} S_N(i) \times 2^i$
    repeat for each $3 \times 3$ neighbourhood

Compute positive ($H_{P_{\mathbf{GLTP}}}$) and negative ($H_{N_{\mathbf{GLTP}}}$) GLTP histograms for $m \times n$ regions
    for each region $(i, j)$ do
        compute positive GLTP histogram: $H_{P_{\mathrm{GLTP}}}(i, j)$
        compute negative GLTP histogram: $H_{N_{\mathrm{GLTP}}}(i, j)$

Concatenate positive ($H_{P_{\mathbf{GLTP}}}$) and negative ($H_{N_{\mathbf{GLTP}}}$) histograms from all $m \times n$ regions to form feature vector
    $H_{P_{\mathrm{GLTP}}}(1,1) H_{N_{\mathrm{GLTP}}}(1,1) \ldots\ldots H_{P_{\mathrm{GLTP}}}(m,n) H_{N_{\mathrm{GLTP}}}(m,n)$
```

3.2.1 Facial Component Extraction

Each region of a face contains varying amounts of discriminative information with regards to facial expression. Regions around the eyes, nose and mouth tend to produce the most discriminative information [77] for appearance based feature extraction methods. However, many appearance-based methods of FER, including GLTP, still populate feature vectors using information obtained from the whole face [36][40][42]. This makes the feature vector unnecessarily large by filling it with redundant data that contains no discriminative expression information; such as that from the edges of the face. In certain cases, the subjects hair or other edge lying obscurities could unintentionally be included as part of the information in the feature vector. This, combined with a large amount of redundant information could have a potentially negative effect on classification accuracy. Results from literature have shown that performing feature extraction on specific facial components cropped from the whole face can improve classification accuracy [39][77]. Two examples of facial component extraction in current literature are given in Figure 3.2a [39] and Figure 3.2b [34] below. Facial component extraction is considered a pre-processing step and is discussed in greater detail in Section 3.3.



(a) Eyes & mouth

(b) Eyes, nose & mouth

Figure 3.2: Extracting facial components from the whole face

3.2.2 Scharr operator

The Sobel-Feldman [76] operator used with GLTP [42] may produce inaccuracies when computing the gradient magnitude of an image. This is because the operator only computes an approximation of the derivatives of the image. While this estimation may be sufficient for most purposes, Scharr [78] proposed a new operator that uses an optimized filter for convolution based on minimizing the weighted mean-squared angular error in the Fourier domain. The Scharr operator is as fast as the Sobel operator but provides much greater accuracy when calculating the derivatives of an image. This should result in a much more accurate representation of the gradient magnitude image. A comparison between the filter masks for a 3 × 3 Sobel & Scharr kernel are shown in Figure 3.3 [79], while a comparison between Sobel and Scharr gradient magnitude images is given in Figure 3.4.

$$\underset{\text{(a)}}{\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}} \qquad \underset{\text{(b)}}{\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}}$$

$$\underset{\text{(c)}}{\begin{bmatrix} -3 & 0 & +3 \\ -10 & 0 & +10 \\ -3 & 0 & +3 \end{bmatrix}} \qquad \underset{\text{(d)}}{\begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ +3 & +10 & +3 \end{bmatrix}}$$

Figure 3.3: Sobel (a) Horizontal & (b) Vertical masks versus Scharr (c) Horizontal and (d) Vertical masks for a 3 × 3 kernel

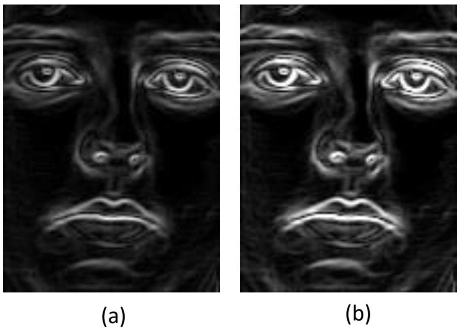


Figure 3.4: (a) Sobel gradient magnitude image versus (b) Scharr gradient magnitude image

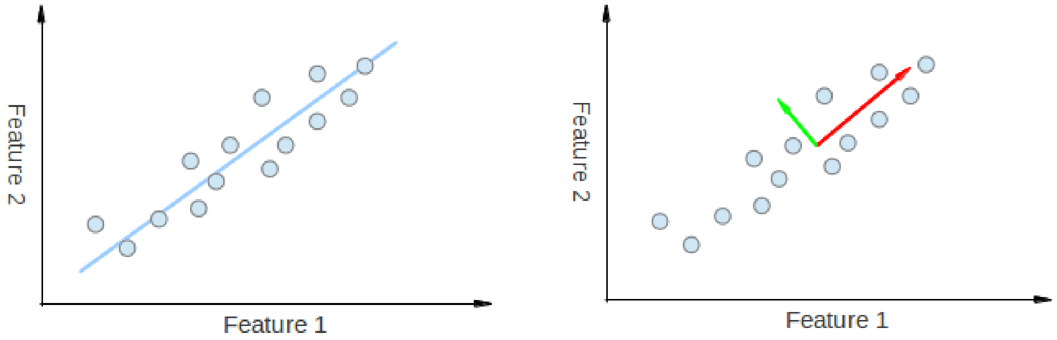
3.2.3 Dimensionality Reduction

Having too few features within a feature vector will most often result in classification failure even when using the best of classifiers. On the other hand, having a very large feature vector will make the classification process slow and is not guaranteed to increase classification accuracy. This is especially true if the feature vector contains large amounts of redundant data. To solve this issue, a dimensionality reduction (DR) [80] technique is proposed to reduce the size of the feature vector; improving classification efficiency without compromising recognition accuracy.

Principal Component Analysis (PCA) [81] is a technique that is used to transform existing features into a newly reduced set of features. PCA has widely been used for face and expression recognition [44][82] with good accuracy and more recently has also been used as a DR technique [34][40]. In [40], both PCA and Adaboost [45] were used for DR and it was shown that PCA outperformed Adaboost. In [42], the same method of FER was performed without DR and lower recognition accuracies were achieved compared to those reported in [40]; demonstrating that DR can improve recognition accuracy as well as efficiency by removing redundant data.

Using PCA, a covariance data matrix is used to compute eigenvectors for a set of data. A linear weighted combination of the topmost few eigenvectors is used to approximate each input feature. All the eigenvectors define the eigenspace and each eigenvalue defines its corresponding axis of variance. Eigenvalues that are close to zero can be discarded as they do not contain much discriminative information. The eigenvectors associated with the top eigenvalues defines the reduced subspace and the original feature vector can be projected onto this subspace to reduce its size; thus achieving dimensionality reduction.

For example, take the 2D set of points in Figure 3.5a [83]. We see that the points could be approximated by the blue line as shown and also note that the points vary the most along the line. Now if we were to perform PCA on the feature space, we would obtain two eigenvectors as shown in Figure 3.5b [83]. The two eigenvectors indicate where the most variation in data occurs; i.e. the principle components. By using principle components to represent the data, we could reduce the dimensionality from 2D to 1D.



(a) 2D set of points

(b) Using principle components to represent the points

Figure 3.5: Principle Component Analysis

The procedure for implementing Improved GLTP is presented in Algorithm 3.2 and illustrated in Figure 3.6. The process is similar to that of GLTP with a few exceptions. Firstly, feature extraction is performed on facial components as opposed to the cropped face. Secondly, the more accurate Scharr gradient operator is used to find the horizontal and vertical derivatives of the source images compared to the Sobel operator used with GLTP. Finally, Principle Component Analysis (PCA) is used to reduce the dimension of the feature vector before classification.

Algorithm 3.2 Improved Gradient Local Ternary Pattern

```
Require: Pre-processed cropped facial components        ▷ see Section 3.3.
Ensure: Feature vector with reduced dimension

1: Initialize variables
   $t$ ←set the threshold

2: Compute horizontal ($G_x$) and vertical ($G_y$) derivative approximations
   of source image using Scharr operator

3: Compute the gradient magnitude for each pixel of the source image
   $G_{x,y} = \sqrt{G_x^2 + G_y^2}$

4: Apply threshold of $\pm t$ around centre gradient value ($G_c$) in a $3 \times 3$
   neighbourhood to determine $S_{\mathbf{GLTP}}$ codes for the gradient magnitude
   image
   for each gradient magnitude value ($G_i$) around $G_c$ do
       if $G_i > G_c + t$ then $S_{\text{GLTP}}(i) \leftarrow +1$
       else if $G_i < G_c - t$ then $S_{\text{GLTP}}(i) \leftarrow -1$
       else$S_{\text{GLTP}}(i) \leftarrow 0$
   repeat for each $3 \times 3$ neighbourhood

5: Compute positive ($P_{\mathbf{GLTP}}$) and negative ($N_{\mathbf{GLTP}}$) GLTP coded image
   representations from $S_{\mathbf{GLTP}}$ values
   for each $S_{\text{GLTP}}(i)$ value in a $3 \times 3$ neighbourhood do
       if $S_{\text{GLTP}}(i) > 0$ then $S_P(i) \leftarrow 1$ & $S_N(i) \leftarrow 0$
       else if $S_{\text{GLTP}}(i) < 0$ then $S_P(i) \leftarrow 0$ & $S_N(i) \leftarrow 1$
       else$S_P(i) \leftarrow 0$ & $S_N(i) \leftarrow 0$
   $P_{GLTP} = \sum_{i=0}^{7} S_P(i) \times 2^i$
   $N_{GLTP} = \sum_{i=0}^{7} S_N(i) \times 2^i$
   repeat for each $3 \times 3$ neighbourhood

6: Compute positive ($H_{P_{\mathbf{GLTP}}}$) and negative ($H_{N_{\mathbf{GLTP}}}$) GLTP histograms
   for $m \times n$ regions
   for each region $(i, j)$ do
        compute positive GLTP histogram: $H_{P_{\text{GLTP}}}(i, j)$
        compute negative GLTP histogram: $H_{N_{\text{GLTP}}}(i, j)$

7: Concatenate positive ($H_{P_{\mathbf{GLTP}}}$) and negative ($H_{N_{\mathbf{GLTP}}}$) histograms from
   all $m \times n$ regions to form extended GLTP histogram
   $H_{P_{\text{GLTP}}}(1,1)H_{N_{\text{GLTP}}}(1,1) \ldots\ldots H_{P_{\text{GLTP}}}(m,n)H_{N_{\text{GLTP}}}(m,n)$

8: Repeat steps 2 to 7 for each remaining facial component & concatenate
   all extended GLTP histograms to form feature vector

9: Apply PCA to feature vector to reduce dimensionality
   produce feature vector with reduced dimension
```

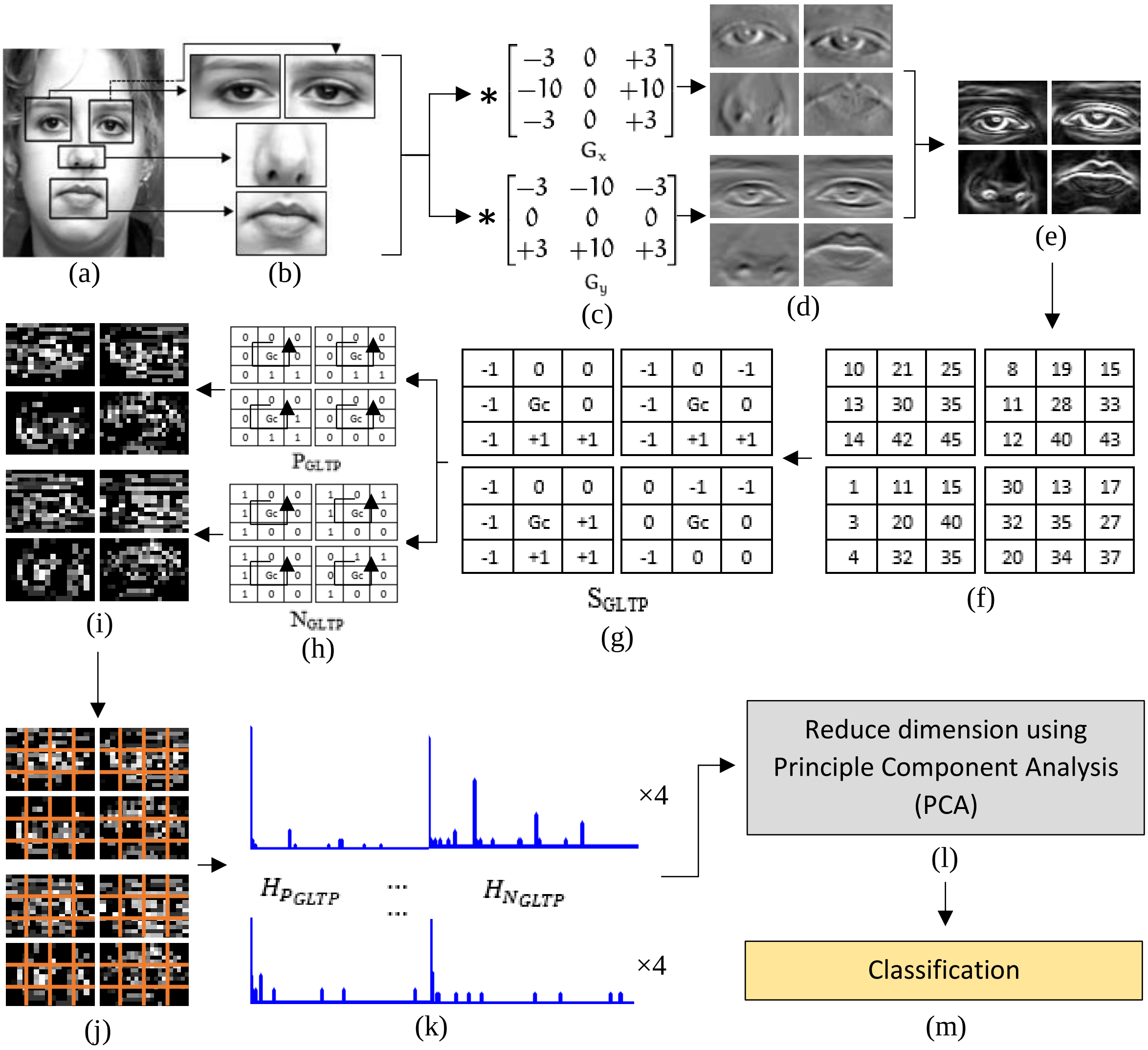


Figure 3.6: The Improved GLTP process (a) Source image (b) Facial component extraction (c) Horizontal & vertical Scharr masks (d) Horizontal and vertical gradient images from Scharr operator (e) Gradient magnitude images (f) Sample 3 × 3 neighbourhoods of gradient magnitude images (g) GLTP codes for $t = 10$ (h) Splitting GLTP code into positive and negative codes (i) Positive and negative decimal coded images (j) Segmenting positive and negative images into 3 × 4 regions (k) Concatenate positive and negative GLTP histogram from each region for each facial component to form feature vector (l) Reduce dimension of feature vector using Principle Component Analysis (PCA) (m) Pass reduced feature vector to classifier

3.3 Pre-processing

The JAFFE [50] and CK/CK+ [51][52] datasets were used to test our proposed methodology (see Section 6.2). To ensure a fair and accurate comparison with results in current literature, images from the datasets were pre-processed before feature extraction and classification. For GLTP, we cropped the face from each image while for Improved GLTP, we cropped multiple facial components from each image. Both methods of pre-processing are discussed in detail below with reference to the CK/CK+ dataset.

3.3.1 Cropped Face

To remove the background and other edge-lying obscurities, the subjects face was cropped from the original image based on the positions of the eyes. For the CK/CK+ dataset, 68 facial landmark locations were provided for each image, each of which represents a point on the face as shown in Figure 3.7 [84]. Using the provided landmarks, the centers of the left and right eye were found and the distance (D) between them was calculated. The face was then cropped using empirically selected percentages of D with the center of the left eye as a reference point as shown in Figure 3.8. In our setup, the cropping region was reduced, compared to [40][42], to exclude non-discriminative expression regions of the face. After cropping, the faces were resized to a uniform size of 147×108 pixels. The pre-processing procedure for GLTP is presented in Algorithm 3.3, followed by a brief explanation.

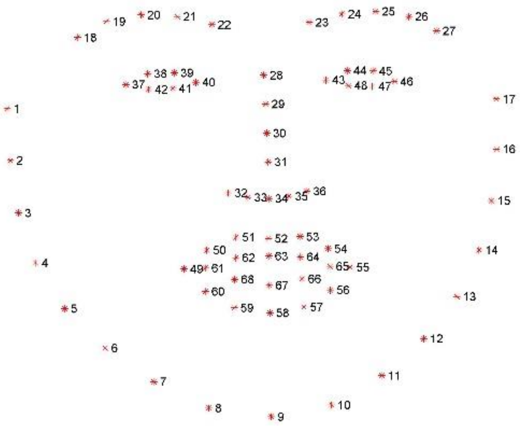


Figure 3.7: 68 facial landmarks

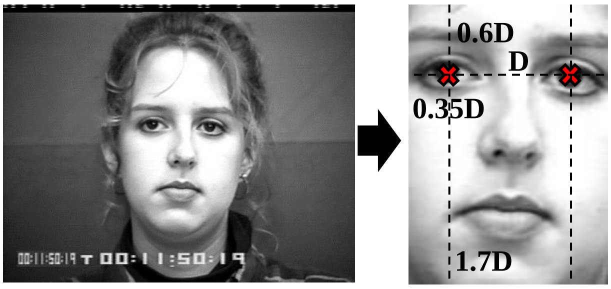


Figure 3.8: Face cropping in the CK/CK+ dataset

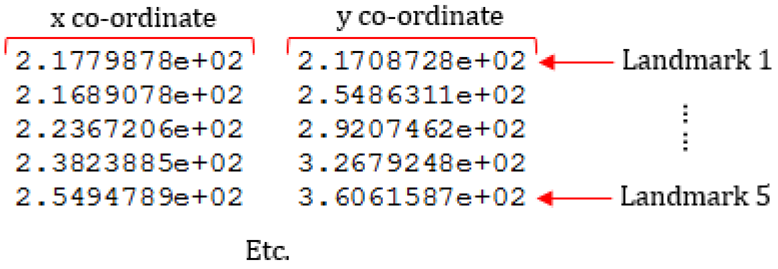
Algorithm 3.3 GLTP pre-processing

Require: 68 facial landmarks for each image
Ensure: Pre-processed cropped faces

```
1: Load images & their corresponding co-ordinates for the corners of the eyes
2: Determine co-ordinates for center of left & right eyes
3: Determine the distance between the centers of the eyes
4: Determine the region of interest (ROI) for the cropped face
5: Crop the ROI from the image
6: Resize the cropped face
7: Repeat steps 2 to 6 for each remaining image
```

Explanation of Algorithm 3.3

1. Load images & their corresponding co-ordinates for the corners of the eyes: We assume that prior to loading the images, a location text file containing the x and y co-ordinates for each of the 68 facial landmarks has been made available for each image. A portion of the location text file is shown in Figure 3.9.



```
2.1779878e+02  2.1708728e+02
2.1689078e+02  2.5486311e+02
2.2367206e+02  2.9207462e+02
2.3823885e+02  3.2679248e+02
2.5494789e+02  3.6061587e+02
```

Figure 3.9: Facial landmark locations inside text file

We open the location text file for each image and read the x & y co-ordinates for the facial landmarks 37, 40, 43 & 46 that represent the corners of the eyes (see Figure 3.7).

2. Determine co-ordinates for center of left & right eyes: To find the center x co-ordinate for each of the eyes, we take the x co-ordinate of the left landmarks 37 & 43 and add half the width of corresponding eye. The center y co-ordinates are represented by the y co-ordinates for the landmarks 40 & 43.

$$x_{\text{left eye center}} = x_{37} + \frac{x_{40} - x_{37}}{2} \tag{3.8}$$

$$y_{\text{left eye center}} = y_{40} \tag{3.9}$$

$$x_{\text{right eye center}} = x_{43} + \frac{x_{46} - x_{43}}{2} \tag{3.10}$$

$$y_{\text{right eye center}} = y_{43} \tag{3.11}$$

3. Determine the distance between the centers of the eyes: To determine the distance **D** between the centers of the eyes, we simply subtract the x co-ordinate of the left eye center from the x co-ordinate of the right eye center as shown:

$$D = x_{\text{right eye center}} - x_{\text{left eye center}} \tag{3.12}$$

4. Determine the region of interest (ROI) for the cropped face: We empirically define a rectangle of size:

$$\text{width} = 1.7 \times D \tag{3.13}$$

$$\text{height} = 2.3 \times D \tag{3.14}$$

$$x_{\text{top left corner}} = x_{\text{left eye center}} - (0.35 \times D) \tag{3.15}$$

$$y_{\text{top left corner}} = y_{\text{left eye center}} - (0.6 \times D) \tag{3.16}$$

as the ROI representing the cropped face.

5. Crop the ROI from the image: We use the rectangle from above to crop the face from the image and save it as a new matrix `cropped_Face`.

6. Resize the cropped face: We resize the `cropped_Face` image to 147 × 108 pixels.

7. Repeat steps 2 to 6 for each remaining image: Repeat the relevant pre-processing procedure for each remaining image until all images have been processed.

3.3.2 Cropped Facial Components

Our first task is to decide what components should be extracted from the image. In [34], cropping was performed on the eyes, nose and mouth regions. However, in [39], cropping was only performed on the eye and mouth regions with the nose being excluded. In our testing, greater recognition accuracy was achieved with the mouth region included. Figure 3.10 shows a comparison between recognition accuracy when using 4 facial components (left-eye, right-eye, nose and mouth) and 3 facial components (left-eye, right-eye and mouth) for feature extraction. The testing was performed for 6 and 7 classes of emotion in the CK/CK+ dataset using 10-fold cross validation with optimal parameters. The results show that higher recognition accuracies were achieved when using 4 facial components for feature extraction i.e. when the nose is included.

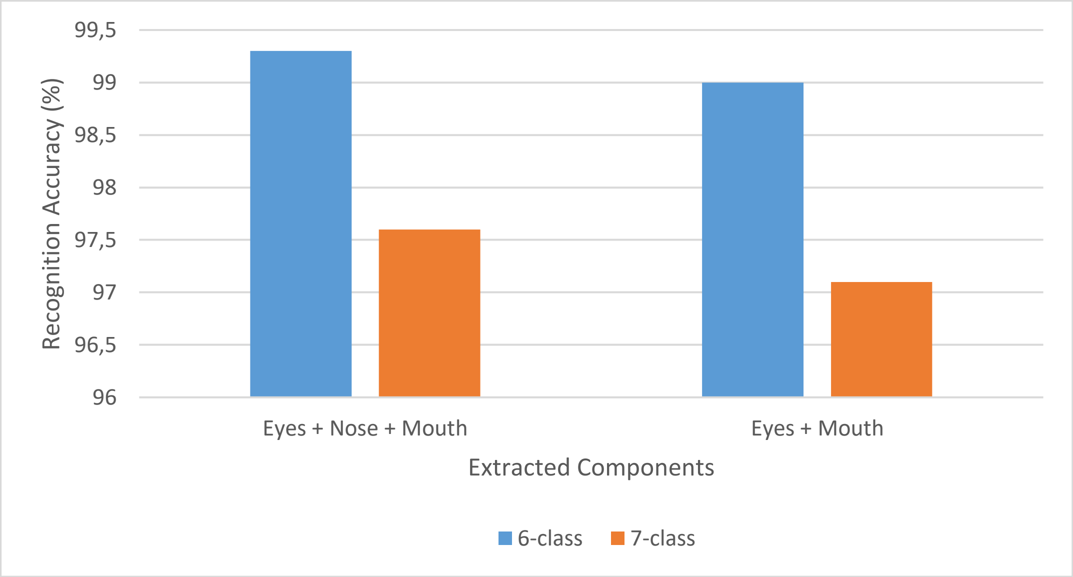


Figure 3.10: The effect of extracting different sets of facial components on recognition accuracy in the CK/CK+ dataset

To determine the optimal number of regions for each component, 10-fold cross validation using optimal parameters was performed for 6 and 7 classes of emotion in the CK/CK+ dataset while varying the region size. The results are shown in Figure 3.11. It can be seen that the optimal region size is 3 × 4 regions. Further increasing the region size past 3 × 4 regions does not result in improved recognition accuracy as unnecessarily large amounts of redundant information gets incorporated into the feature vector degrading accuracy.

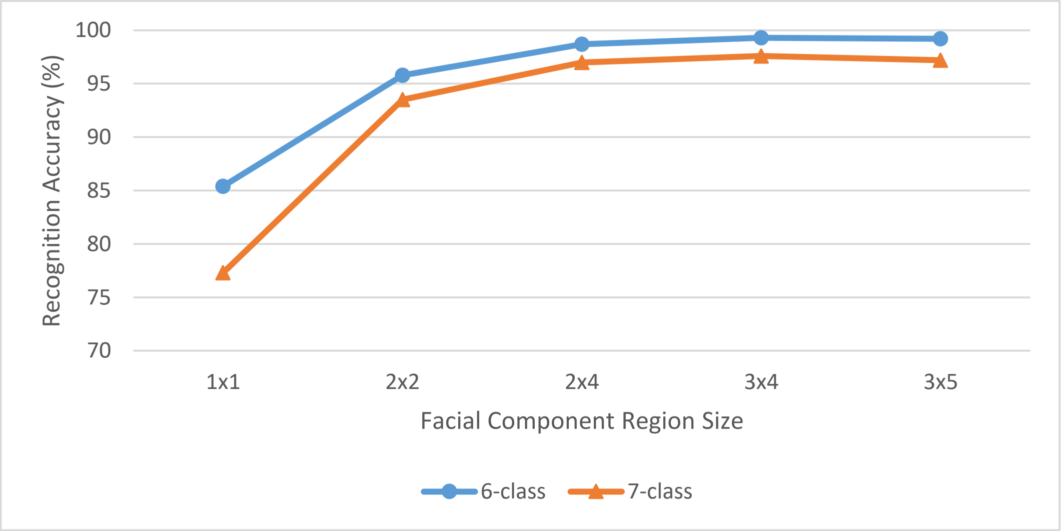


Figure 3.11: The effect of varying facial component region size on recognition accuracy in the CK/CK+ dataset

For the CK/CK+ dataset, the 68 facial landmark locations (see Figure 3.7) provided with each image were used to crop the left eye, right eye, nose and mouth components from the image as shown in Figure 3.12a. After cropping, each component was resized to a uniform size of 75×120 pixels and segmented into 3×4 regions as shown in Figure 3.12b. The pre-processing procedure for Improved GLTP is presented in Algorithm 3.4.

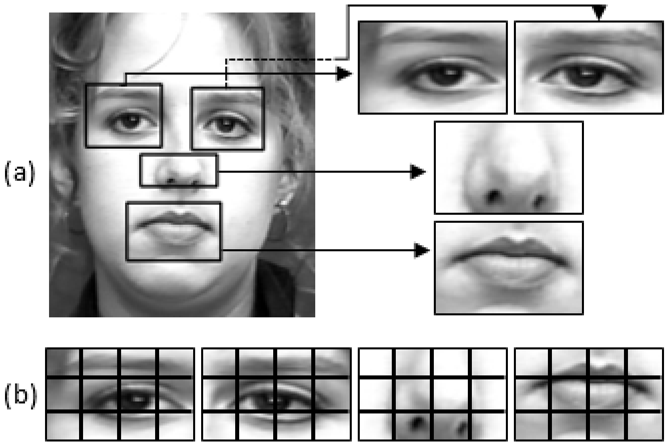


Figure 3.12: Cropped facial components (a) Cropping eyes, nose & mouth components (b) Segmenting each component into a 3×4 region

Algorithm 3.4 Improved GLTP pre-processing

```
Require: 68 facial landmarks for each image
Ensure: Pre-processed cropped facial components
Load images & their corresponding co-ordinates for the left eye, right eye, nose & mouth
Determine region of interest (ROI) for left eye, right eye, nose & mouth
Crop each ROI separately from the image to form individual facial components
Resize each facial component to a uniform size
Repeat steps 2 to 4 for each remaining image
```

The pre-processing procedure for Improved GLTP is similar to that of GLTP (see Algorithm 3.3) with a few notable changes. Firstly, in step 1, we load the co-ordinates for both eyes, the nose and the mouth as opposed to just the eyes in GLTP. Next, we determine the regions of interest for the left eye, right eye, nose and mouth separately and crop each as a separate component. Finally, we resize each component to a uniform size of 75 × 120 pixels.

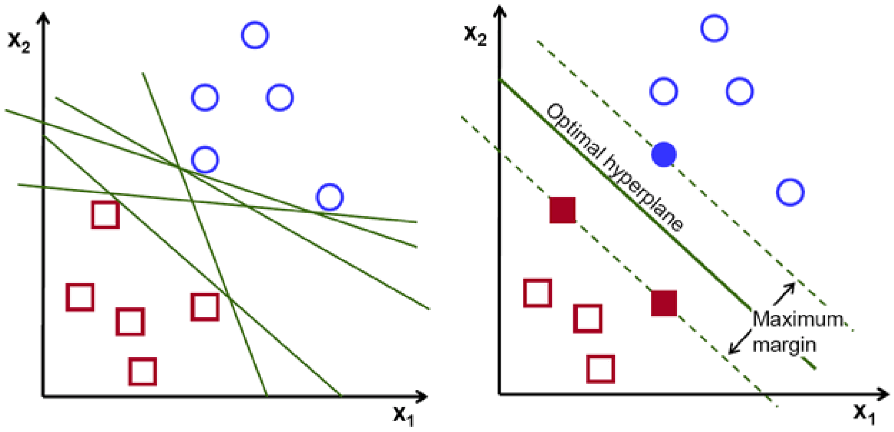
3.4 Classification using Support Vector Machines

A Support Vector Machine (SVM) [46] was used for feature classification. SVM is a supervised machine learning technique that implicitly maps labeled training data into a higher dimensional feature space; constructing a linearly separable optimal hyper-plane between the data. The hyper-plane is said to be optimal when the separating margin between the sample classes is maximal. The optimal-hyperplane can then be used to classify new examples. Given a set of labeled training data: $T = \{(x_i, l_i), i = 1, 2, \ldots, L\}$, where $x_i \in R^P$ and $l_i \in \{-1, 1\}$, a new test sample x is classified by:

$$f(x) = sign(\sum_{i=1}^{L} \alpha_i l_i K(x_i, x) + b) \tag{3.17}$$

where α_i are Lagrange multipliers of the dual optimization problem, $K(x_i, x)$ is a kernel function and b is a bias or threshold parameter. The training samples (x_i) with $\alpha_i > 0$ are the support vectors. The hyper-plane maximizes the margin between these support vectors.

Consider a simplified example of a linearly separable set of 2D points belonging to two different classes in the Cartesian plane as shown in Figure 3.13 [85]. We can see that there are many possible lines that could be drawn to separate the data as shown in Figure 3.13a. Lines that pass too close to the points are considered bad as they will not generalize well and over-fitting will occur. The goal of SVM is thus to find the optimal separating hyperplane which maximizes the the minimum distance between the classes of data. For this example, the optimal hyperplane is shown in Figure 3.13b.



(a) Multiple separating lines (b) Finding the optimal hyperplane

Figure 3.13: Support Vector Machine

SVM is traditionally a binary classifier that constructs an optimal hyper-plane from positive and negative samples. For multi-class classification, the one-against-rest approach was employed. In this approach, a binary classifier is trained for each expression to discriminate one expression from the rest. A radial basis function (RBF) kernel was used for classification. The RBF kernel is defined by the equation:

$$K(x_i, x) = exp(-\gamma \|x_i - x\|^2), \gamma > 0 \tag{3.18}$$

where γ is a user selectable kernel parameter. The procedure for finding optimal parameters for classification is discussed in Section 6.3.2.

3.5 Chapter Summary

In this chapter, improvements to the robust feature selection method Gradient Local Ternary Pattern (GLTP) [42] were proposed. Facial component extraction was used to extract regions of the face containing the most discriminative expression information, such as the eyes, nose and mouth. This allowed for much of the redundant expression information from the edges of the face, previously included with GLTP, to be excluded from feature selection. For computation of the horizontal and vertical derivatives of the images, the more accurate Scharr gradient operator was used compared to the Sobel operator used with GLTP. This resulted in a more accurate representation of the gradient magnitude image which better reflects the true texture of the images. Because the number of features extracted from the images tends to be very large, Principle Component Analysis (PCA) was used to reduce the dimension of the feature vectors, further removing any redundant information before classification. Together the improvements form the method Improved GLTP. The pre-processing methods for GLTP & Improved GLTP were detailed and the use of a Support Vector Machine for classification was discussed. In Section 6.3, we perform an in depth comparison between the experimental results of Improved GLTP and GLTP, as well as other state of the art methods in current literature.

Chapter 4

Crowd Emotion Dataset

4.1 Introduction

To test the performance and accuracy of our proposed Crowd Monitoring system we require a dataset of crowd images with ground-truth emotions based on the facial expressions within the crowd. With the recent interest in Crowd Monitoring techniques, many crowded scene datasets have become publicly available for training and testing. A few of the most commonly cited crowd datasets [8] are summarized in Table 4.1.

Table 4.1: Popular crowded scene datasets

Dataset	**Size**	**Scenario**	**Environment**	**Ground-truth**
UCF Behaviour [86]	61 seq.	behaviour	outdoor	yes
UMN [87]	11 videos	panic	indoor/outdoor	yes
UCSD [88]	98 videos	abnormality	outdoor	yes
CUHK [18]	2 videos	traffic/pedestrian	outdoor	no
Violent-Flows [89]	246 videos	violence	outdoor	yes

The UCF Behaviour [86] dataset consists mainly of image sequences from the web representing crowds with various behaviours; such as panic, fight, congestion etc. The UMN [87] dataset contains videos of escape/panic events in both indoor and outdoor environments where crowds are trying to leave an area at once. The UCSD [88] dataset contains videos taken by a camera located outdoors above pedestrian walkways. Abnormalities are marked by non pedestrian entities in the walkways and anomalous pedestrian motion patterns. The CUHK [18] dataset contains two videos; one for traffic and another for pedestrians. The pedestrian video was filmed outdoors in a subway station however contains no ground-truth annotations for crowd behaviour or abnormalities. The Violent-Flows [89] dataset contains real-world video footage of crowd violence collected from YouTube.

Two of the datasets, UCSD and CUHK, are aimed towards testing abnormality detection techniques. However only the UCSD dataset contains annotations for abnormal events. The remaining three datasets are aimed at analyzing crowd behaviour. The UMN and Violent-Flows datasets were created specifically for panic and violent behaviours whereas the UCF Behaviour dataset contains image sequences of various crowd behaviours. However, none of these datasets are aimed at emotion detection in crowds. Although emotion can be linked to behaviour [20], behaviour alone is not descriptive enough to represent distinct emotions. In terms of the camera placement and quality, the datasets mentioned here are also not suitable for identifying facial expressions. Although there are many datasets readily available for Facial Expression Recognition (FER), such as the JAFFE [50] and CK/CK+ [51][52] datasets, each image contains a closeup of only one subject's face. We thus propose creating a new dataset to bridge the gap between crowded scene analysis and FER.

Our proposed Crowd Emotion dataset consists of crowd images; with the individuals in the crowd expressing certain facial expressions based on one of seven prototypic emotions [5]. Rather than capturing images of crowds expressing different emotions, we propose capturing individual full-body images of multiple subjects expressing different facial expressions; then overlaying them into a background scene to create a crowded setting. The advantage in using this method is that we can create different subsets of crowd emotion images on demand without having to put a group of people together. For example, we can easily vary the size of the crowd or specifically set the type of emotion being expressed in different areas of the crowd. The ground-truth emotion of the crowd is dependent on the type of emotions being expressed within the crowd. The images of the individual subjects are included along with the crowd images and their ground-truth emotions.

One could argue that, because crowds are very dynamic in nature, a video analysis would be more suitable for emotion detection compared to still images. The main purpose for video analysis of crowds, however, is to monitor crowd movement patterns not emotion. In scenes of non-panic, crowd movement may be unreliable for inferring emotion and does not help to distinguish between multiple types of similar emotion in crowds. While a dynamic method of Facial Expression Recognition (FER) could be used to detect the evolution of emotion over time, the computational cost will be high and may not be suitable for real-time application. In terms of FER, the evolution of facial expressions may also not be available in full sequences as individuals may turn away from the camera at certain points in time. For FER, still images represent a suitable form of media to detect crowd emotion at different points in time and if analyzed in real-time, will still contain dynamic elements of the crowd.

In reality, the emotion of a crowd has many dynamics and can be represented by body posture, facial expression and even movement. In this work, we focus on using Facial Expression Recognition (FER) for emotion analysis. In this case, the assumption is made that crowd emotion is expressed solely through facial expression. Thus the ground-truth emotion of the crowd images is set based on the facial expressions present within the crowd and not any other form of emotion expression such as body posture. To ensure that results from testing are consistent, all faces within the crowd are unobscured and frontal facing; simulating a crowd in optimal conditions. It must be noted that, in real world conditions, faces in a crowd may be highly obscurred or facing away from the camera. These challenges are discussed further in the final chapter of this work.

Our Crowd Emotion dataset consists of 14 students enrolled at the University of KwaZulu-Natal. The students are between 21 to 24 years of age. Of the 14 students, 9 were male and 5 were female. The subjects were asked to express six prototypic emotions (i.e. joy, surprise, anger, fear, disgust and sadness) as well as a neutral emotion in three different sessions to account for any variation. The images were captured in an indoor environment using a fixed camera positioned in front of the subjects. The images were digitized to 1280×720 pixels in Portable Network Graphics (PNG) image format. The creation of the crowd images consists of three main steps; (1) Capturing the subject images, (2) Extracting the subjects, & (3) Overlaying the subjects to create a crowded scene. All three steps are discussed in detail in Sections 4.2 to 4.4. We aim to publish the dataset online for public use so that in the future, the dataset can be used in other work such as facial recognition, people counting, and face detection in crowded scenes. All of the subjects involved in the formulation of the dataset gave their consent for the use of their images in publications (see Appendix A).

4.2 Capturing the subject images

The first step in creating the Crowd Emotion dataset was to capture full-body still images of each subject expressing different emotions. The camera was positioned in front of the subjects and was fixed in place; one image of the background was captured. Each subject was then asked to express 7 different emotions starting with a neutral emotion and ending with a sadness emotion; one image was captured for each emotion. This process was repeated 3 times for each subject to account for any variation in the way they expressed each emotion. The images were saved as PNG files using the directory structure shown in Figure 4.1.

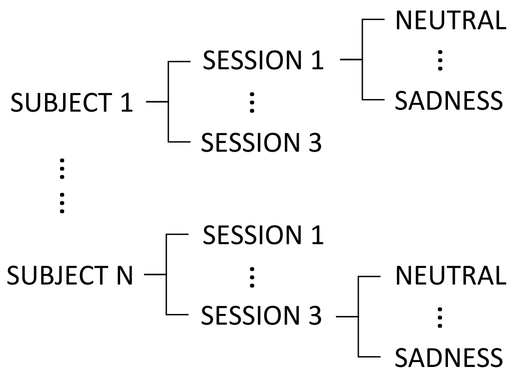
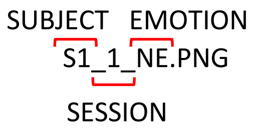


Figure 4.1: Directory structure for saving subject images

Each image was named using the following format:



where NE = Neutral, HA = Joy, SU = Surprise, AN = Anger, FE = Fear, DI = Disgust & SA = Sadness. A sample of the directory structure and output images after capturing emotions for the 14 subjects is shown in Figure 4.2 below.

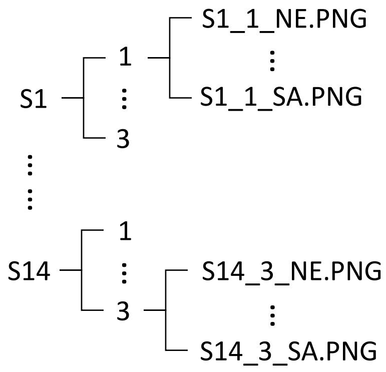


Figure 4.2: Directory structure and output images for 14 subjects

4.3 Extracting subjects from the images

After having captured images for each subject, the next step is to extract the subject out of the image. This is achieved by creating a mask for each image that distinguishes between the subject and the background. The steps for performing this task are shown in Algorithm 4.1 and are explained in detail after.

Algorithm 4.1 Extracting subjects from images

```
Require: Subject images, Background image
Ensure: Cropped subject images & corresponding subject mask

1: Initialize variables
   $threshold \leftarrow$ set threshold for background subtraction

2: Load background & subject images

3: Perform background subtraction
   produce threshold image

4: Crop the subject image & threshold image

5: Find the largest contour within the cropped threshold image

6: Create mask for subject by filling in the contour

7: Output cropped subject image & corresponding mask

8: Repeat steps 3 to 7 for each image
```

1. Initialize variables: We empirically set the threshold value for background subtraction based on the lighting condition within the room. This is discussed further in Step 3.

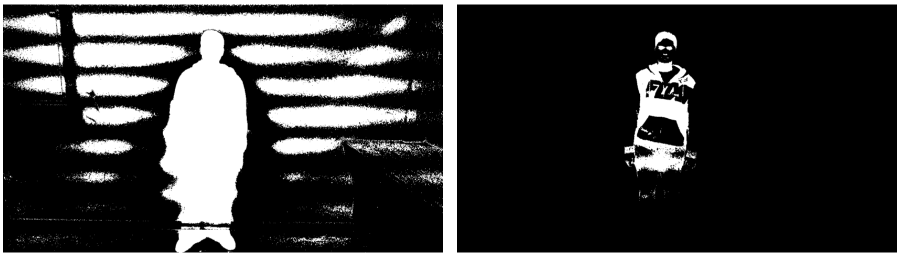
2. Load background & subject images: We load each of the subject images and the background image. Both the background image and subject image are required for background subtraction.

3. Perform background subtraction: In this step, we perform background subtraction to isolate the subject from the background. The background image is subtracted from the subject image and the absolute difference is compared to the set *threshold* value. If the difference is above the threshold it is considered part of the foreground, else it is considered part of the background. This process is summarized in Equation 4.1 below.

$$G(x,y) = \begin{cases} 1, & \text{if } |I(x,y) - B(x,y)| > \text{threshold} \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

where $I(x,y)$ is the subject image, $B(x,y)$ is the background image and $G(x,y)$ is the resultant threshold image (with 1 representing the foreground & 0 representing the background).

The *threshold* value should be set so that as little possible noise & illumination are present in the threshold image. Figure 4.3 shows the result of setting the threshold too low and too high. In Figure 4.3a, too much illumination and noise are included as part of the foreground whereas in Figure 4.3b, parts of the foreground are mistaken for the background.

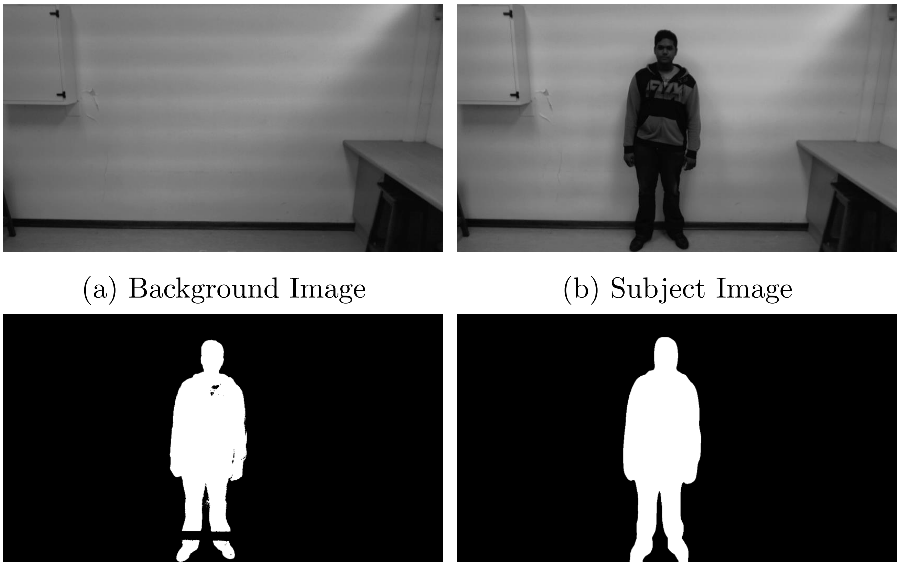


(a) Threshold too low (b) Threshold too high

Figure 4.3: Incorrectly setting the threshold for background subtraction

After having empirically set the threshold to an optimal value, we perform background subtraction to obtain the threshold image. A *blur* function is applied to the threshold image to smooth the image and remove any noise around the subject. The result is shown in Figure 4.4.

4. Crop the subject image & threshold image: The subject image and threshold image are uniformly cropped to remove the unnecessary parts of the background; leaving only the subject centered in the frame. The region of interest for cropping is determined empirically and the result is shown in Figure 4.5.



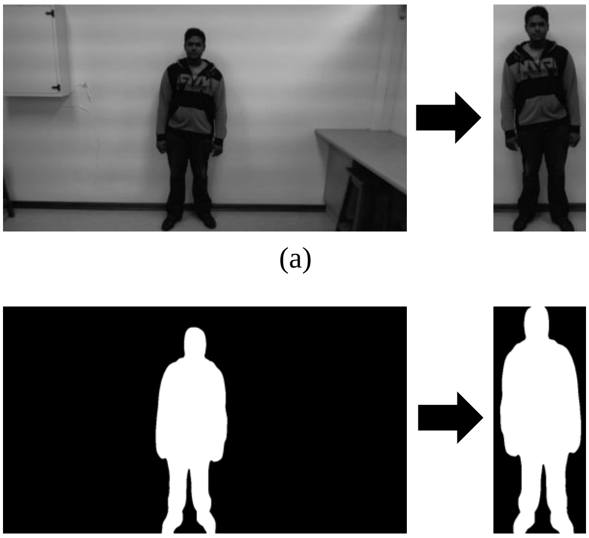
(a) Background Image

(b) Subject Image

(c) Threshold Image

(d) Blur Function

Figure 4.4: Result of background subtraction



(a)

(b)

Figure 4.5: Cropping the (a) subject image & (b) threshold image to a uniform size

5. Find the largest contour within the cropped threshold image: We identify the outline of the subject by finding the largest contour that exists within the smoothed threshold image. By finding the largest contour, we eliminate the chance of accidentally including any noise present within the image in the mask. In the example image above, no noise is present; thus there exists only a single contour as shown in Figure 4.6.

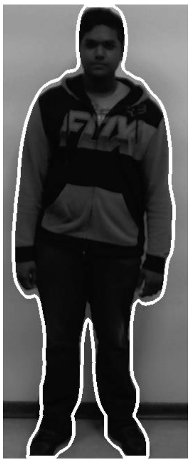


Figure 4.6: Finding the contour of the subject within the image

6. Create mask for subject by filling in the contour: To create the subject mask, we simply set all points within the interior of the contour to 1 and all points outside of the contour to 0. The non-zero values of the mask represent the subject. By finding and filling in the contour, we ensure that no holes exist within the mask. As there are no holes and no noise present within the threshold image after applying the *blur* function in Figure 4.4d; the smoothed threshold image and mask are, in this case, equivalent.

7. Output cropped subject image & corresponding mask: The cropped subject image and its corresponding mask are written to file. They will both be used in Section 4.4.

8. Repeat steps 3 to 7 for each image: The relevant steps are repeated for each subject image.

4.4 Creating crowd images with emotion

The final step is to place the cropped subject images together in another frame using their corresponding masks. The steps for performing this task are shown in Algorithm 4.2, followed by a detailed explanation of each step.

Algorithm 4.2 Creating crowd images with emotion

```
Require: Cropped subject images & their masks, Image of empty environment
Ensure: Crowd image with emotion
  Load the cropped subject images & their corresponding image masks
  Load an image of an empty environment
  Identify regions of interest (ROI) to place subjects in the chosen environment
  Copy one subject image to each ROI using its corresponding mask
```

1. Load the cropped subject images & their corresponding image masks: The subject images to be used depends on the type of crowd that is to be created. For example, if the crowd is to only contain violent and neutral emotions, then only the subject images exhibiting violent and neutral emotions should be loaded.

2. Load an image of an empty environment: An image of a suitable empty environment is sourced from the web. The image could be an indoor or outdoor environment as long as it does not contain any other individuals within it. The environment we used is shown in Figure 4.7.

Figure 4.7: Empty indoor environment for placing subjects

3. Identify regions of interest (ROI) to place subjects: Areas where subjects are to be placed are identified within the empty environment. The regions of interest should be of the same size as the subject images and are dependent on the number of subjects present in the crowd & whether or not occlusions of the face should exist. In Figure 4.8, regions of interest are identified such that the crowd contains the maximum number subjects as possible without facial occlusions.

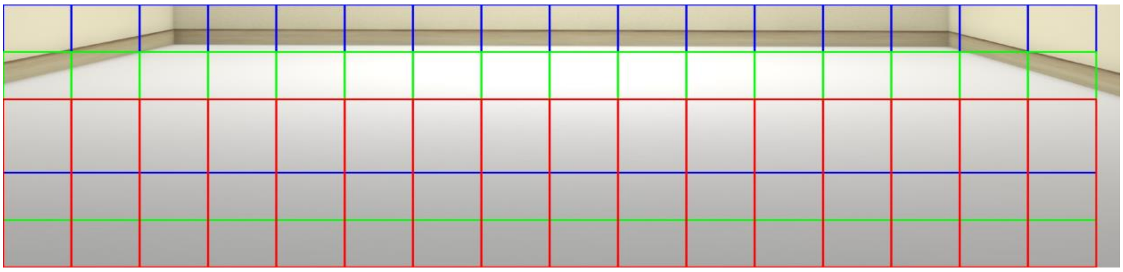


Figure 4.8: Identifying regions of interest to place subjects within the chosen environment

4. Copy one subject image to each ROI using its corresponding mask: Finally, we place one subject in each ROI starting with the top row as shown in Figure 4.9. This is done by copying the cropped subject image to the ROI using its corresponding mask. The non-zero elements of the mask indicate which elements of the subject image should be copied. By starting at the top row and working our way down, the subjects are overlaid on top of one another in a way that simulates a crowded scene. The end result of randomly selecting subjects expressing neutral emotion is shown in Figure 4.10. The size of the crowd can also be adjusted as shown in Figure 4.11. The ground-truth emotion of the crowd can be based off of the dominant emotions present within the crowd (see Section 6.5).

4.5 Chapter Summary

In this chapter, we investigated current datasets used for Crowd Monitoring and found that they were not suitable for testing our proposed system. We thus proposed and detailed the creation of a novel Crowd Emotion dataset by combining individual subjects expressing different types of emotion through facial expression in the same scene. The dataset is used to test the effectiveness of face detection in Section 6.4 and crowd emotion estimation in Section 6.5.

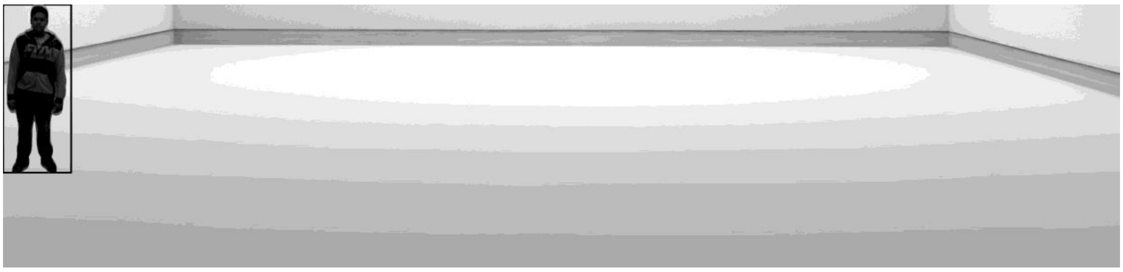


Figure 4.9: Placing a subject in the ROI using its corresponding mask

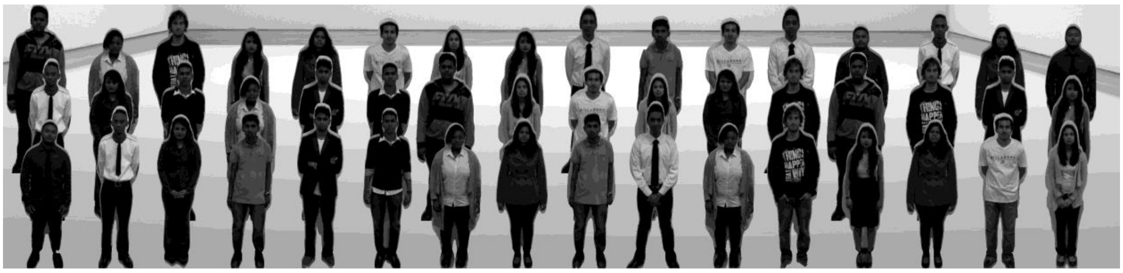


Figure 4.10: Crowd of neutral emotion

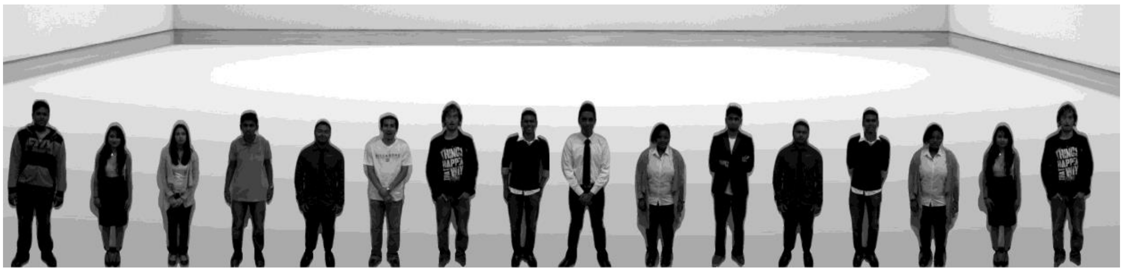


Figure 4.11: Crowd reduced in size

Chapter 5

Emotion Estimation in Crowds

5.1 Introduction

This chapter presents methodology for estimating the overall emotion of a crowd. Firstly, in Section 5.2, a robust and accurate technique for face detection is presented. Section 5.3 makes note of the use of Facial Expression Recognition (FER) to predict the emotion being expressed in each of the detected faces. In Section 5.4, the distance between the detected faces is computed using a fully connected, undirected graph. In Section 5.5, each face's closest neighbour is found using a Minimum Spanning Tree. Finally, Section 5.6 presents a method for estimating crowd emotion based on the size and weighting of groups of similar emotion in the crowd.

5.2 Face Detection

Over the years, many different techniques have been proposed for face detection. One such method is that of Active Shape Models (ASM) [35]; which is commonly used together with FER to identify facial points and edges in the pre-processing stage [36][37]. The Viola and Jones [33] face detection algorithm, which uses a boosted cascade of classifiers to rapidly detect faces, has been shown to be extremely effective at identifying faces in uncontrolled backgrounds with great accuracy [90] compared to other existing face detection techniques. In our work, the Viola and Jones method was selected for face detection due to its combination of speed and accuracy. One real-world limitation with this method is that it is mostly designed to detect frontal faces; however, for our purposes this is not a limitation but rather an advantage. Faces that are obscured or rotated away from the camera will likely reduce the accuracy of FER; thus we require a face detection algorithm that is able to identify only the faces that are "suitable" for feature extraction i.e. those that are mostly frontal facing. The Viola and Jones face detection algorithm consists of three main steps; (1) Computing the integral image, (2) Learning classifiers using Adaboost, and (3) Combining the classifiers in a cascade structure.

1. Computing the integral image: Images are classified using simple features as opposed to pixel intensities. The simple features used are reminiscent of Haar Basis functions and consist of two, three and four rectangle features. Because the set of rectangle features can be very large, the images are first represented by an integral image. The integral image at location (x, y) represents the sum of the pixels above and to the left of (x, y), inclusive:

$$ii(x, y) = \sum_{x\prime \leq x, y\prime \leq y} i(x\prime, y\prime) \tag{5.1}$$

where $ii(x, y)$ is the integral image and $i(x, y)$ is the original image. By using the integral image, the time taken to compute the rectangular feature set at any scale or location is greatly reduced because any rectangular sum can be computed using just four array references.

2. Learning classifiers using Adaboost: The number of rectangle features associated with each image sub-window is far greater than the number of pixels. To ensure fast classification, only a small subset of these features are combined to form an effective classifier. Adaboost [45] is used in such a way that each weak learning algorithm selects only a single rectangle feature which best separates the positive and negative examples. For each of these features, the optimal threshold classification function is computed such that the minimum number of examples are misclassified. A weak classifier $h_j(x)$ is thus represented by:

$$h_j(x) = \begin{cases} 1, & \text{if } p_j f_j(x) < p_j \theta_j \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

where f_j is a feature, θ_j is the threshold, p_j is a parity indicating the direction of the inequality and x is a 24×24 pixel sub-window of an image.

3. Combining the classifiers in a cascade structure: To speed-up the classification process, successively more complex classifiers are combined in a cascade structure. Each stage in the cascade is constructed by training a classifier using Adaboost with the threshold adjusted to minimize false negatives. By using a cascade of classifiers, sub-windows that are not of interest can be quickly discarded in the early stages so that increased computation is spent only on more promising face-like regions in the later stages; greatly increasing the overall computational efficiency of classification. This process is illustrated in Figure 5.1, where a negative outcome at any of the increasingly complex classifier stages leads to the immediate rejection of the sub-window.

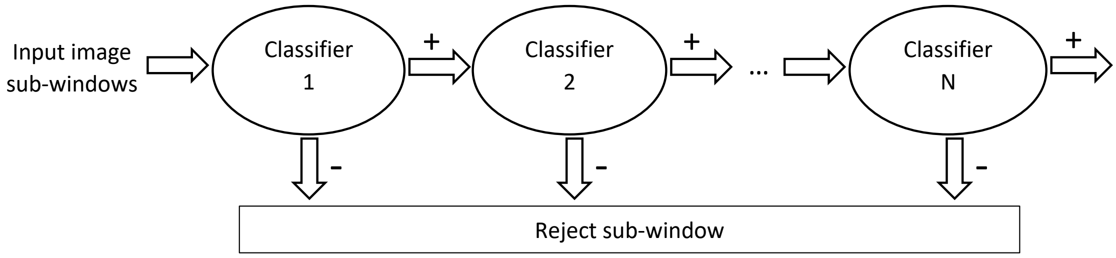


Figure 5.1: Cascading weak classifiers to improve overall performance

We investigate the effectiveness of the Viola and Jones face detection algorithm in Section 6.4.

5.3 Facial Expression Recognition (FER)

Facial Expression Recognition (FER) has already been discussed in detail in Chapter 3. At this point we simply note that each of the detected faces in the crowd undergoes pre-processing, facial feature extraction and classification to identify the emotion being expressed by each individual. The use of individual emotion for crowd emotion estimation is discussed later in this chapter.

5.4 Computing the distance between faces

To determine the distance between neighbouring faces; each face is treated as a node, where the vertex of the node is represented by the top left point of the ROI representing the face as shown in Figure 5.2. As in [91], a **fully-connected undirected graph** is used to link every node's vertex with one another, where the distance between any two nodes is represented by the weight of the connecting edge. We say the resulting graph is **fully-connected** because each node is connected to every other node present, and **undirected** because there is only one unique edge between each pair of nodes (direction does not matter). As such, for N nodes we have a total of $(N \times (N-1))/2$ edges; where the distance between nodes i and j is found using the Euclidean norm as:

$$\text{Distance}_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{5.3}$$

where (x_i, y_i) represents the vertex of node i and (x_j, y_j) represents the vertex of node j. The graph is represented by an $N \times N$ adjacency matrix (Adj_Mat), where $\text{Adj\_Mat}_{i,j} = \text{Distance}_{i,j}$. The algorithm for finding the fully-connected undirected graph is given in Algorithm 5.1. The fully-connected undirected graph for a group of 4 people is shown in Figure 5.2. The weight of each edge represents the Euclidean distance between the nodes.

Algorithm 5.1 Finding a fully-connected undirected graph of distance

```
Require: Number of Nodes (N), Nodes
Ensure: Adjacency matrix (Adj_Mat)

1: Initialize variables
   Adj_Mat ← 0
   Distance ← 0

2: Find distance between nodes
   for each node $i \leftarrow 0$ to $N-1$ do
       for each node $j \leftarrow i+1$ to $N-1$ do
           Distance ← $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$
           $\text{Adj\_Mat}_{i,j}$ ← Distance
           $\text{Adj\_Mat}_{j,i}$ ← Distance
```

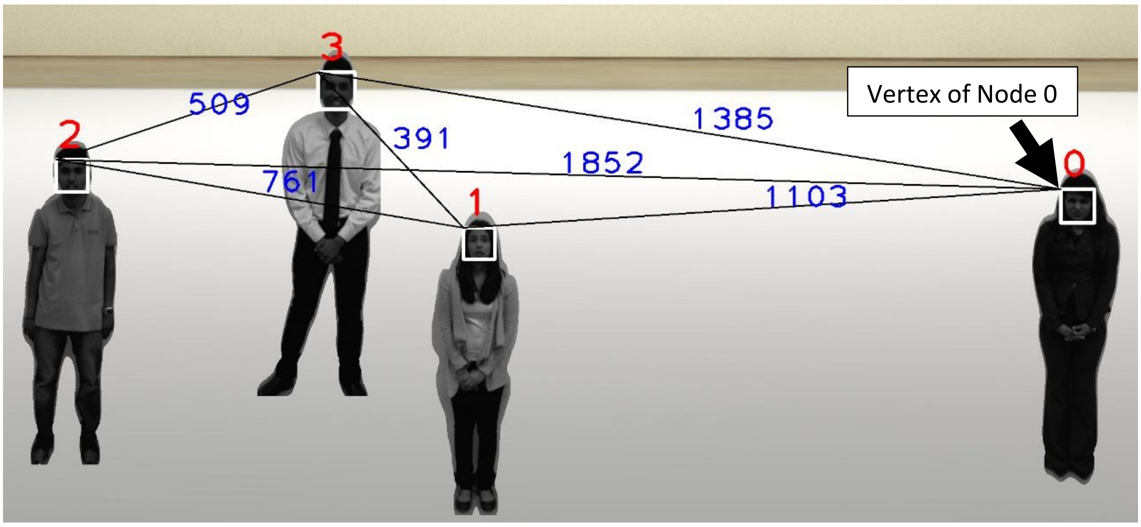


Figure 5.2: Fully-connected undirected graph of distance for a small crowd

The adjacency matrix for the fully-connected undirected graph of the crowd above is given below.

$$\text{Adj\_Mat} = \begin{pmatrix} 0 & 1103 & 1852 & 1385 \\ 1103 & 0 & 761 & 391 \\ 1852 & 761 & 0 & 509 \\ 1385 & 391 & 509 & 0 \end{pmatrix}$$

Increasing the crowd size from 4 to 20 people exponentially increases the number of edges present in the graph from 6 to 190 edges as shown in Figure 5.3.

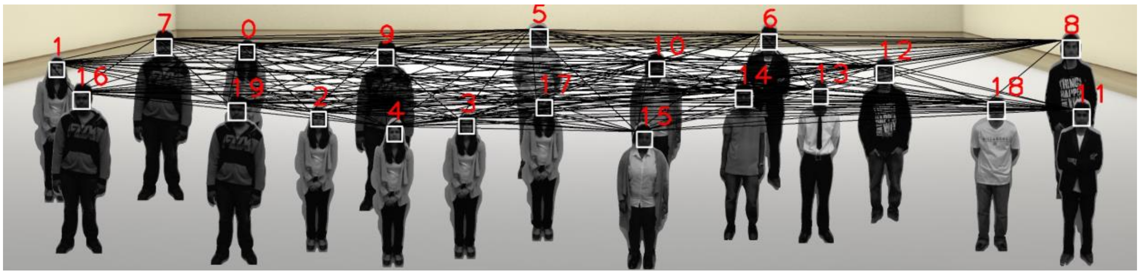


Figure 5.3: Fully-connected undirected graph for a larger crowd

5.5 Computing the closest neighbours of each face

A **Minimum Spanning Tree (MST)** is used to represent each face's closest neighbours as suggested in [91]. A spanning tree of a graph G is a tree, where every edge in the tree belongs to G and, that includes every node of G. The cost of a spanning tree is represented by the sum of the weights of all edges in the tree. A MST is a spanning tree where the cost is a minimum. Numerous approaches have been suggested for finding a MST. Two of the most popular approaches are Kruskal's algorithm and Prim's algorithm [92]. In this work, Prim's algorithm was used to find the MST. Starting with an empty MST, for each step of Prim's algorithm, we consider a group of edges that connects the set of nodes already included in the MST with the set of nodes not yet included. The edge with minimum weight is selected and the node is added to the MST. The procedure is repeated until all nodes have been included in the MST. Prim's algorithm, which takes a fully-connected undirected graph (represented by an adjacency matrix) as the input and outputs a vector of parent nodes and their children (representing the MST), is summarized in Algorithm 5.2.

The time complexity for Prim's algorithm using an adjacency matrix is $\mathcal{O}(N^2)$, where N represents the total number of nodes in the graph. By representing the graph as an adjacency list, rather than an adjacency matrix, and using a min heap; the time complexity can be reduced to $\mathcal{O}((N+E)\log N)$, where E represents the total number of edges in the graph. That is; it takes $\mathcal{O}(N+E)$ time to traverse all nodes in the graph using an adjacency list and $\mathcal{O}(\log N)$ time for operations on the

Algorithm 5.2 Finding a Minimum Spanning Tree (MST)

```
Require: Adjacency matrix (Adj_Mat)
Ensure: Vector (Parent) representing the completed MST

1: Initialize variables
   MST ← add node 0
   $Parent_0$ ← -1 (set node 0 as the root node)

2: Compute the MST

   while MST does not include all nodes do

       Find a node $v$ that is not included in the MST with a minimal weighting
       edge connecting to a node $u$ included in the MST

       Add node $v$ to MST

       $Parent_v \leftarrow u$
```

min heap.

The MST for the fully-connected undirected graph of the crowd given in Figure 5.3 is shown below. In a MST there is a total of $N-1$ edges.

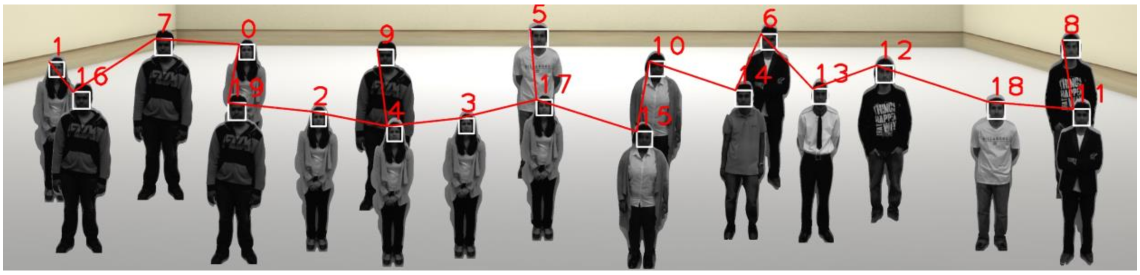


Figure 5.4: Minimum Spanning Tree for a crowd of 20 people

The MST shown in Figure 5.4 can be represented using a vector shown in Figure 5.5.

$$\begin{array}{rcccccccccccccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 \\ \text{Parent} = & (-1 & 16 & 19 & 4 & 2 & 17 & 14 & 0 & 11 & 4 & 15 & 18 & 13 & 6 & 10 & 17 & 7 & 3 & 12 & 0\,) \end{array}$$

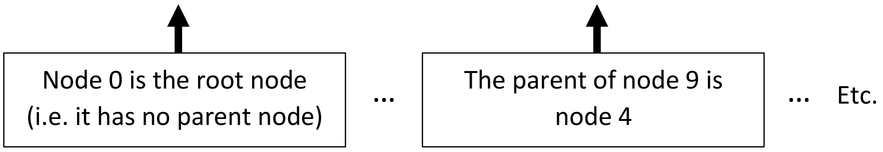


Figure 5.5: Vector representation of the Minimum Spanning Tree

5.6 Estimating crowd emotion from groups of individual emotion

To estimate the overall emotion of a crowd, we employ both FER and a method to identify groups of individuals who are expressing the same emotion and who are situated close together in the crowd. These groups of individuals can be represented by chains of emotion, where the length of each chain is represented by the number of individuals in the chain. The overall emotion of the crowd can then be estimated by finding the largest chain of emotion with the greatest weighting. This approach is more accurate at estimating crowd emotion compared to more simplistic methods such as finding the predominant individual emotion in the crowd. Because individuals in a crowd can take on the emotion of the people around them, it is possible that even a relatively small group of individuals expressing one emotion can influence the emotion of the individuals around them who in turn can influence the individuals around them. This chain reaction is known as the Domino effect and can potentially lead to crowds getting out of control. Our proposed crowd emotion estimation technique aims to identify sufficiently large groups of individuals with the same emotion, such as anger, before it is able to spread any further. This allows for early detection of potentially problematic crowds.

This section is split into two parts; firstly, finding chains of emotion in the crowd and secondly, using the largest chains of emotion with the greatest weighting to predict the overall crowd emotion. The algorithm for finding chains of emotion in the crowd using it's MST and the predicted emotion of each face is given in Algorithm 5.3. The emotion chains can be represented by a matrix (chain_Emotion) of size $N \times N$, where the row represents the start node of the chain, the column represents the end node of the chain and chain_Emotion$_{start,end}$ = length of chain$_{start,end}$.

Algorithm 5.3 Finding chains of emotion in the crowd

```
Require: Vector (Parent) representing the MST, The predicted emotion of each
    node
Ensure: Matrix (chain_Emotion) representing the chains of emotion in the crowd

1: Initialize variables
   $start \leftarrow$ set starting node to Node 0
   $end \leftarrow$ ending node
   $parent \leftarrow$ set parent node to Node 0
   $child \leftarrow$ child node
   $counter \leftarrow$ set initial chain count to 1
   $branches \leftarrow$ vector to store remaining branch nodes
   chain_Emotion $\leftarrow 0$

2: Check if parent node has a child
   if parent has a child that has not yet been checked then
   a: Mark the child node
        $child \leftarrow$ child of $parent$
   else
   b: End chain
        $end \leftarrow parent$
        $\text{chain\_Emotion}_{start,end} \leftarrow counter$
        $counter \leftarrow 0$
   c: Check for any remaining branches
      if branches is not empty then goto step 5
      else end

3: Check if parent node has another child
   if parent has another child that has not yet been checked then
   d: Save branch for later
        add child to branches vector
   e: End chain
        $end \leftarrow parent$
        $\text{chain\_Emotion}_{start,end} \leftarrow counter$
        $counter \leftarrow 0$
   f: Start new chain
        $start \leftarrow parent$

4: Check if the emotion of the child is the same as the parent
   if $\text{emotion}_{child} = \text{emotion}_{parent}$ then
   g: Continue chain
        increment counter
   else
   h: End chain
        $end \leftarrow parent$
        $\text{chain\_Emotion}_{start,end} \leftarrow counter$
        $counter \leftarrow 0$
```

Algorithm 5.3 Finding chains of emotion in the crowd (continued)

```
    i: **Start new chain**
         $start \leftarrow child$
         increment $counter$
    j: **Child becomes the parent**
    $parent \leftarrow child$
    goto step 2

5: **Go back to unfinished branch**
    remove child from $branches$ and set as $child$
    $parent \leftarrow \text{Parent}_{child}$
    k: **Start new chain**
    **if** $\text{emotion}_{child} = \text{emotion}_{parent}$ **then** $start \leftarrow parent$
    **else** $start \leftarrow child$
    goto step 3
```

Consider the crowd in Figure 5.4. The emotion chains for the crowd are illustrated in Figure 5.6, where the values above each node represent the node number and predicted FER emotion label of the node. There are a total of 2 unique emotion chains in the crowd; one with emotion label 0 (Anger) and another with emotion label 4 (Neutral). The starting nodes for these unique emotion chains are indicated in Figure 5.6 using black arrows. The resultant emotion chains in the crowd can be represented by the chain_Emotion matrix given in Figure 5.7. In the following steps (1 to 17), the chain_Emotion matrix was constructed.

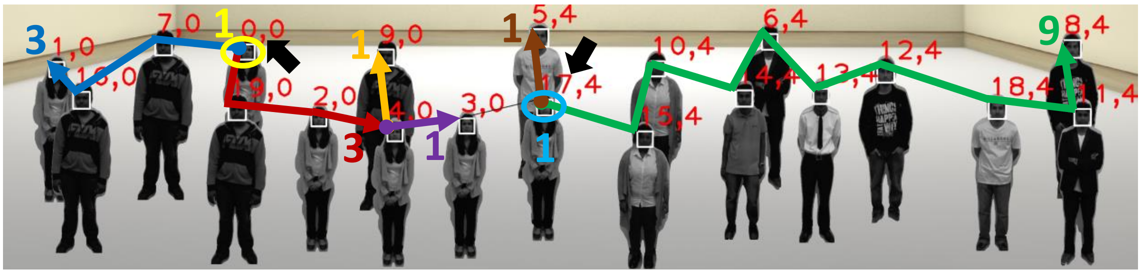


Figure 5.6: Finding chains of emotion in the crowd

1. The starting node and parent node are set to Node 0. $Counter = 1$.
2. Check if Node 0 has a child. It does and Node 7 is marked as the child node (Node 19 could also be marked instead).
3. Check if Node 0 has any other children. It does and Node 19 is saved for later. End the current chain and set $\text{chain\_Emotion}_{0,0} = 1$. A new chain is started with Node 0 as the starting node. $Counter$ is reset.

4. Check if child Node 7 has the same emotion as parent Node 0. It does, so continue the chain by incrementing the *counter*. Node 7 now becomes the parent and the process is repeated.

5. The chain continues until the parent node becomes Node 1 ($counter = 3$). Node 1 does not have a child node, so end the chain and set chain_Emotion$_{0,1} = 3$. *Counter* is reset.

6. Go back to the unfinished branch with parent Node 0 and child Node 19. The emotion of Node 19 is the same as that of Node 0 so set the start of the new chain to Node 0 and increment the *counter*.

7. The chain continues until Node 4 ($counter = 3$). Node 4 has multiple children, so mark Node 3 as the child and save Node 9 for later. End the chain and set chain_Emotion$_{0,4} = 3$. A new chain is started with Node 4 as the starting node. *Counter* is reset.

8. Child Node 3 has the same emotion as parent Node 4, so continue the chain by incrementing the *counter*. Node 3 becomes the parent node.

9. The child of Node 3 is Node 17 which has a different emotion, so end the chain, set chain_Emotion$_{4,3} = 1$, and start a new chain with the Node 17 as the starting node. $Counter = 1$.

10. Node 17 becomes the parent and has multiple children, so mark Node 5 as the child and save Node 15 for later. End the chain ($counter = 1$) and set chain_Emotion$_{17,17} = 1$. A new chain is started with Node 17 as the starting node. *Counter* is reset.

11. Child Node 5 has the same emotion as parent Node 17, so continue the chain by incrementing the *counter*. Node 5 becomes the parent node.

12. Node 5 does not have a child node, so end the chain ($counter = 1$) and set chain_Emotion$_{17,5} = 1$. *Counter* is reset.

13. Go back to the unfinished branch with parent Node 4 and child Node 9. The emotion of Node 9 is the same as that of Node 4 so set the start of the new chain to Node 4 and increment the *counter*.

14. Node 9 does not have a child node, so end the chain ($counter = 1$) and set chain_Emotion$_{4,9} = 1$. *Counter* is reset.

15. Go back to the unfinished branch with parent Node 17 and child Node 15. The emotion of Node 15 is the same as that of Node 17 so set the start of the new chain to Node 17 and increment the *counter*.

16. The chain continues until the parent node becomes Node 8 ($counter = 9$). Node 8 does not have a child node, so end the chain and set chain_Emotion$_{17,8} = 9$. *Counter* is reset.

17. No unfinished branches remain so all chains of emotion have been found.

$$
\begin{array}{cc|cccccccccccccccccccc}
 & & \multicolumn{20}{c}{\text{End}} \\
 & & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 \\
\hline
 & 0 & \textcircled{1} & \textcircled{3} & 0 & 0 & \textcircled{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 4 & 0 & 0 & 0 & \textcircled{1} & 0 & 0 & 0 & 0 & 0 & \textcircled{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{Start} & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 17 & 0 & 0 & 0 & 0 & 0 & \textcircled{1} & 0 & 0 & \textcircled{9} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \textcircled{1} & 0 & 0 \\
 & 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

Figure 5.7: Matrix representing the chains of emotion in the crowd

To predict the overall emotion of the crowd; the largest total length of each emotion chain must first be found. The size of each emotion chain in relation to the crowd is then compared to a set threshold value, *thresh*, which represents the minimum size required for the chain to be considered large enough to influence the overall crowd emotion. Each prototypic emotion is assigned a weighting representing its importance. In our work, all emotions are assigned an equal weighting with the exception of Neutral emotion which is assigned a lower weighting. This is because Neutral emotion does not provide much information about the emotional state of the individuals within the crowd. The overall crowd emotion is predicted as the emotion belonging to the chain that meets the following requirements:

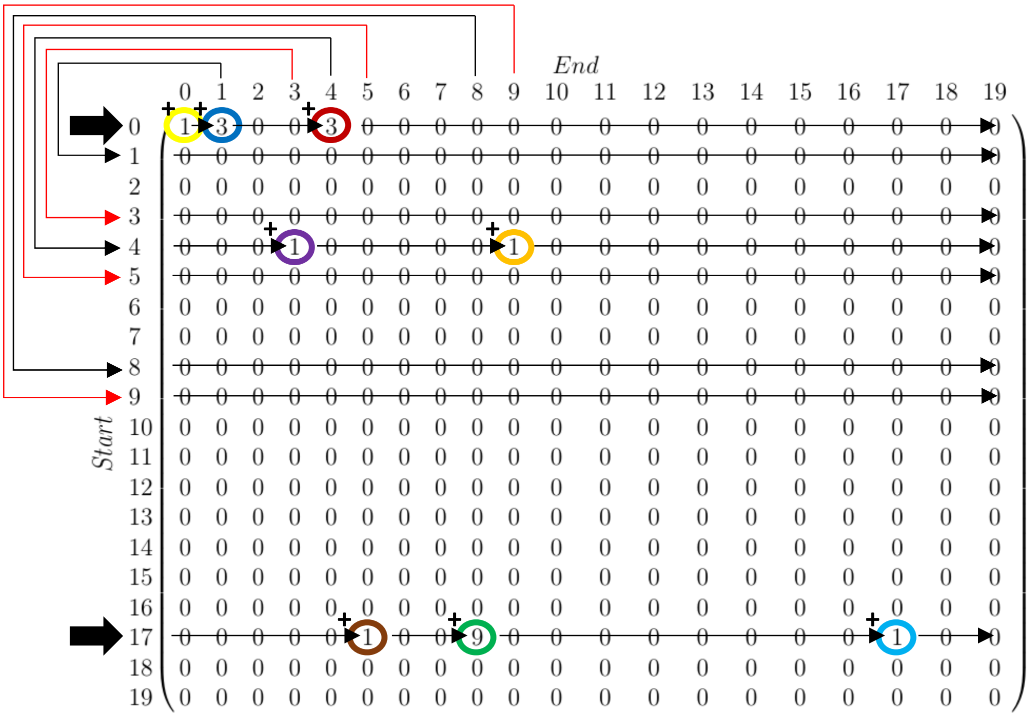
1. The size of the chain in relation to the crowd is greater than or equal to *thresh*.
2. The emotion of the chain has the greatest possible weighting out of the chains that meet requirement (1).
3. The size of the chain is the largest out of the chains that meet requirements (1) and (2).

If no chain meets the above requirements; the emotion of the crowd is considered to be mixed. The algorithm for predicting the overall emotion of the crowd is summarized in Algorithm 5.4.

Algorithm 5.4 Predicting the overall emotion of the crowd

```
Require: Matrix (chain_Emotion) representing the chains of emotion in the crowd,
    Weighting of each emotion, Number of Nodes (N)
Ensure: Predicted crowd emotion

1: Initialize variables
    $start \leftarrow$ starting node of the chain
    $end \leftarrow$ ending node of the chain
    $counter \leftarrow$ used to count the total length of a chain of emotion
    $max\_count \leftarrow$ vector to store the largest length of each emotion chain
    $per \leftarrow$ the percentage size of the emotion chain in terms of the whole crowd
    $thresh \leftarrow$ min. % required for the emotion chain to be considered large enough
    $crowd\_emotion \leftarrow$ the predicted emotion of the overall crowd

    function addChains($start$, $counter$)
        for each node $end \leftarrow 0$ to $N - 1$ do
            $counter \leftarrow counter + \text{chain\_Emotion}_{start,end}$
            if $\text{chain\_Emotion}_{start,end}$ is non-zero and $start \neq end$ then
                addChains($end$, $counter$)        ▷ end node becomes the start node
        return $counter$

2: Find the largest total length of each emotion chain
    for each starting node belonging to a unique emotion chain do
        $counter \leftarrow 0$
        $start \leftarrow$ starting node
        $counter \leftarrow$ addChains($start$, $counter$)
        if $counter > max\_count_{\text{emotion}_{start}}$ then $max\_count_{\text{emotion}_{start}} \leftarrow counter$

3: Sort max_count in descending order of chain length

4: Sort max_count in descending order of emotion weighting

5: Compute the overall emotion of the crowd
    for each emotion chain $e$ do
        $per \leftarrow (max\_count_e/\text{total nodes}) \times 100\%$
        if $per \geq thresh$ then
            $crowd\_emotion \leftarrow$ emotion $e$
            end
    if $crowd\_emotion$ is not set then $crowd\_emotion \leftarrow$ mixed
```

Consider the chains of emotion in Figure 5.7. The procedure for predicting the emotion of the crowd using *thresh* = 30% is given below.



$$\begin{array}{c|cccccccccccccccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 \\ \hline
0 & 1 & 3 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
17 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

Figure 5.8: Finding the largest total length of each emotion chain

1. Refer to Figure 5.8. There are 2 unique starting nodes, Node 0 and Node 17, which represent the start of 2 unique chains of emotion in the crowd. The first step is to find the total length of each unique emotion chain. Working with Node 0 as the starting node (row 0 of the matrix chain_Emotion) and *counter* = 0, check to see if any chains exist with *start* Node 0. A chain exists between Node 0 and itself with length 1. Add 1 to the *counter* (*counter* is now 1) and continue as *start* = *end*. Move on to the chain between Node 0 and Node 1 with length 3. Add 3 to the *counter* (*counter* is now 4) and check to see if there are any branches to this chain by temporarily setting the *start* node to 1 (*end* node becomes the *start* node). Row 1 of the matrix is empty (there are no branch chains) so return to row 0. A chain exists between Node 0 and Node 4 with length 3. Add 3 to the *counter* (*counter* is now 7) and check row 4 to see if there are any branch chains. A chain exists between Node 4 and and Node 3 with length 1. Add 1 to the *counter* (*counter* is now 8) and check row 3. Row 3 is empty so return to row 4. A chain also exists between Node 4 and Node 9 with length 1. Add 1 to the *counter* (*counter* is now 9) and check row 9. Row 9 is empty so once again return to row 4. There are no more chains with *start* Node 4 so return to row 0 where we left off. There are also no more chains with *start* Node 0 so set the total chain length of the emotion of the starting node, which in this case is **Anger**, to the value of the *counter*, which is **9**. Repeat the same steps for the other unique starting Node 17. The total chain length of **Neutral** emotion is $(1+9+1) =$ **11**. Only the largest chain length for each emotion is saved.

2. The emotion chains are sorted in descending order of chain length. In this case we have **Neutral** ← 11 then **Anger** ← 9.

3. The emotion chains are sorted in descending order of emotion weighting. In this case we have **Anger** ← 9 then **Neutral** ← 11. The emotion chains are now sorted by weighting and length.

4. The percentage size of each emotion's chain length in terms of the whole crowd is computed using the equation:

$$per = \frac{\text{total length of emotion chain}}{\text{total crowd size}} \times 100\% \tag{5.4}$$

   Firstly, we have $per_{\text{Anger}} = 45\%$ which is greater than the required threshold value of $thresh = 30\%$; so the overall emotion of the crowd is predicted to be **Anger**. We note that $per_{\text{Neutral}} = 55\%$. Although per_{Neutral} is greater than per_{Anger}, the weighting of Neutral emotion is lower than that of Anger emotion; hence why the overall emotion of the crowd was predicted to be Anger, not Neutral.

Consider a crowd where no chain exists with size greater than or equal to $thresh = 30\%$ as shown in Figure 5.9. In this case, the overall emotion of the crowd is mixed.

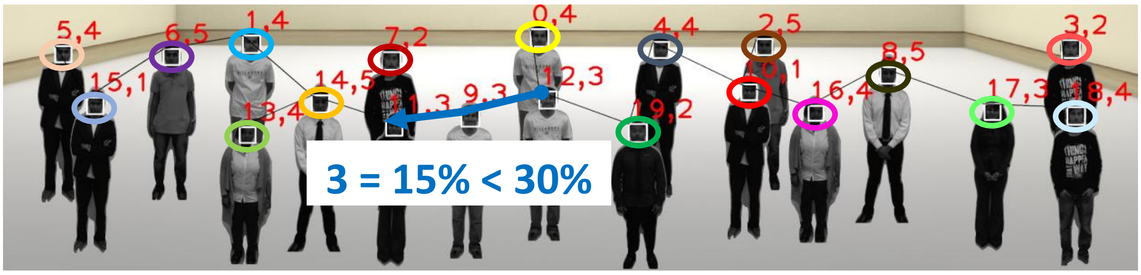


Figure 5.9: Crowd with mixed emotion

5.7 Chapter Summary

In this chapter, methodology for predicting the overall emotion of a crowd was presented in detail. Firstly, the Viola and Jones face detection algorithm was used to identify faces in the crowd so that FER could be used to predict the emotion of each individual. Next, the distances between the individuals were found using a fully connected undirected graph. A Minimum Spanning Tree was then used to find each individuals closest neighbour. Finally, the overall emotion of the crowd was predicted by finding a suitably large group of individuals close together who were all expressing the same emotion. The process for predicting the overall emotion of a crowd is illustrated in Figure 5.10.

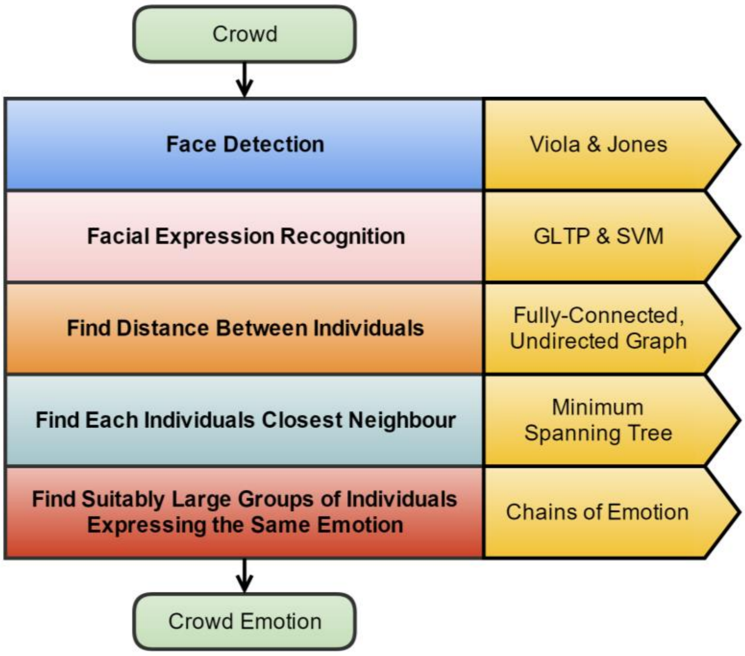

Crowd
Face Detection
Viola & Jones
Facial Expression Recognition
GLTP & SVM
Find Distance Between Individuals
Fully-Connected, Undirected Graph
Find Each Individuals Closest Neighbour
Minimum Spanning Tree
Find Suitably Large Groups of Individuals Expressing the Same Emotion
Chains of Emotion
Crowd Emotion

Figure 5.10: Predicting the overall emotion of a crowd

Our approach to crowd emotion estimation is tested in detail in Section 6.5.

Chapter 6

Experimental Results & Discussion

6.1 Testing Procedures

We selected three different testing procedures to measure the accuracy of our proposed methodology as outlined in [34][53]. The procedures for each form of testing are outlined below.

6.1.1 Cross-Validation (CV)

In k-fold cross-validation, the entire dataset is randomly divided into k roughly equally-sized segments. In our setup, we used $k = 10$ as in [34][42]. For each fold, 9/10 of the segments are used as the training set while the remaining segment is used as the testing set. This process is repeated 10 times so that each segment is tested once. The average accuracy is calculated across the 10-folds. Because the dataset is randomly divided, the average accuracy calculated will be different each time 10-fold cross-validation is run. To ensure fair results, 10-fold cross-validation is performed 10 times and the average accuracy is calculated from all 10 runs.

6.1.2 Leave-One-Out (LOO)

In the leave-one-out method, all images in the dataset apart from one are used as the training set. The remaining image is used for testing. This process is repeated for every image in the dataset so that each image is tested once. The average overall recognition accuracy is found using the equation:

$$\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{total number of images}} \times 100\% \tag{6.1}$$

6.1.3 Person Independent (PI)

In the person independent method, all images apart from one subjects set of images are used as the training set (i.e. one subject is completely left out of the training set). The one remaining subjects images are used as the test set. The process is repeated for each subject. The accuracy from all subjects' tests are averaged to obtain the overall average recognition accuracy.

All three testing procedures actually represent a different form of cross-validation. It is how the folds are selected that sets them apart. In cross-validation, each test fold is random and roughly equally sized. In leave-one-out, each test fold consists of only a single image. In person independent, each test fold consists of only one subject's images.

For testing Facial Expression Recognition (FER) all three testing procedures were used to thoroughly test the method under consideration, allowing for better comparison of recognition accuracies between methods. The general procedure for implementing all three testing procedures with regards to FER is presented in Algorithm 6.1, followed by a brief explanation of each step. For testing our proposed crowd emotion estimation algorithm, only the 10-fold cross-validation procedure was used. Leave-one-out and person independent testing procedures would not be suitable for testing on a crowded environment and thus are excluded. The procedure for testing our crowd emotion estimation algorithm is discussed further in Section 6.5.

Algorithm 6.1 Testing Procedure for Facial Expression Recognition (FER)

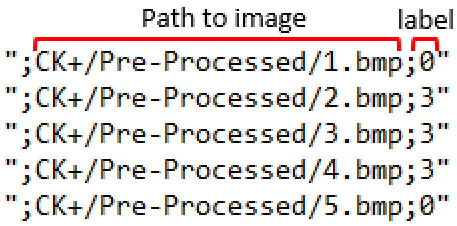
```
 1: Pre-processing                                                        ▷ see Section 3.3

 2: Initialize variables
    vector of images & corresponding vector of labels
    vector of $percentages$, $total$, $average$

 3: Load pre-processed images & their corresponding labels
    return vector of images & vector of their corresponding labels

 4: Randomize the images and their corresponding labels                   ▷ ONLY for CV
    create a vector of indices the same size as images
    randomize the indices
    for each randomized index do
        add corresponding image & label to new vectors; rand_images & rand_labels

 5: Set number of images to be used for training
    $num \leftarrow$ number of training images

 6: Move images & their corresponding labels for testing to end of vector
    for each image $i \leftarrow 0$ to size of images $- num$ do
        move $image(i)$ to end of $rand\_images$ vector
        move $label(i)$ to end of $rand\_labels$ vector

 7: Create training data & train classifier
    for each image $\leftarrow 0$ to $num - 1$ do
        Feature extraction                                                ▷ see Section 3.2
    Train classifier                                                      ▷ see Section 3.4

 8: Create test data & predict
    $count \leftarrow 0$
    for each image $\leftarrow num$ to size of images $- 1$ do
        Feature extraction                                                ▷ see Section 3.2
        Perform prediction                                                ▷ see Section 3.4
        if prediction is correct then increment $count$

 9: Determine recognition accuracy for the fold
    $percentage \leftarrow (count/(\text{size of images} - num)) \times 100\%$
    add $percentage$ to $percentages$ vector

10: Repeat steps 5 to 9 for each remaining fold

11: Determine average recognition accuracy across all folds
    $total \leftarrow$ sum of all $percentages$
    $average \leftarrow total$ / (number of folds)
```

Explanation of Algorithm 6.1

1. Pre-processing: The images must be pre-processed prior to performing tests. For GLTP [42] & Improved GLTP, see Section 3.3.

2. Initialize variables: We initialize vectors that will store the images and their corresponding labels. In terms of tracking recognition accuracy, we create a vector, `percentages`, to store the recognition accuracy for each fold. The variables `total` and `average` will be used at the end to compute the average overall recognition accuracy across all folds.

3. Load pre-processed images & their corresponding labels: The pre-processed images and their corresponding labels for testing are loaded from a Comma Separated Values (CSV) file that contains both the path to the image and its corresponding class label. An example of the entries within the CSV file is given in Figure 6.1.



```
";CK+/Pre-Processed/1.bmp;0"
";CK+/Pre-Processed/2.bmp;3"
";CK+/Pre-Processed/3.bmp;3"
";CK+/Pre-Processed/4.bmp;3"
";CK+/Pre-Processed/5.bmp;0"
```

Figure 6.1: Example entries in CSV file

4. Randomize the images and their corresponding labels (ONLY for CV): This step only needs to be performed for the cross-validation procedure. For the leave-one-out procedure, we only select one image for testing per fold; thus there is no need for randomization as the order in which the images are arranged does not matter. On the other hand, for the person independent procedure it is very important that we do NOT perform randomization. For PI testing, the entries in the CSV file should be arranged by subject.

Assuming we use CV, if we were to randomize both the `images` vector & `labels` vector separately, we would run into problems as the image and label of corresponding indices would no longer match due to different randomizations. To overcome this, we instead create a vector of indices, the same size as the `images` & `labels` vectors, and randomize the indices using a randomization function. We then add each image and corresponding label to new vectors, `rand_images` & `rand_labels`, based on the randomized indices. This results in a randomized vector of images while still maintaining a vector of corresponding labels.

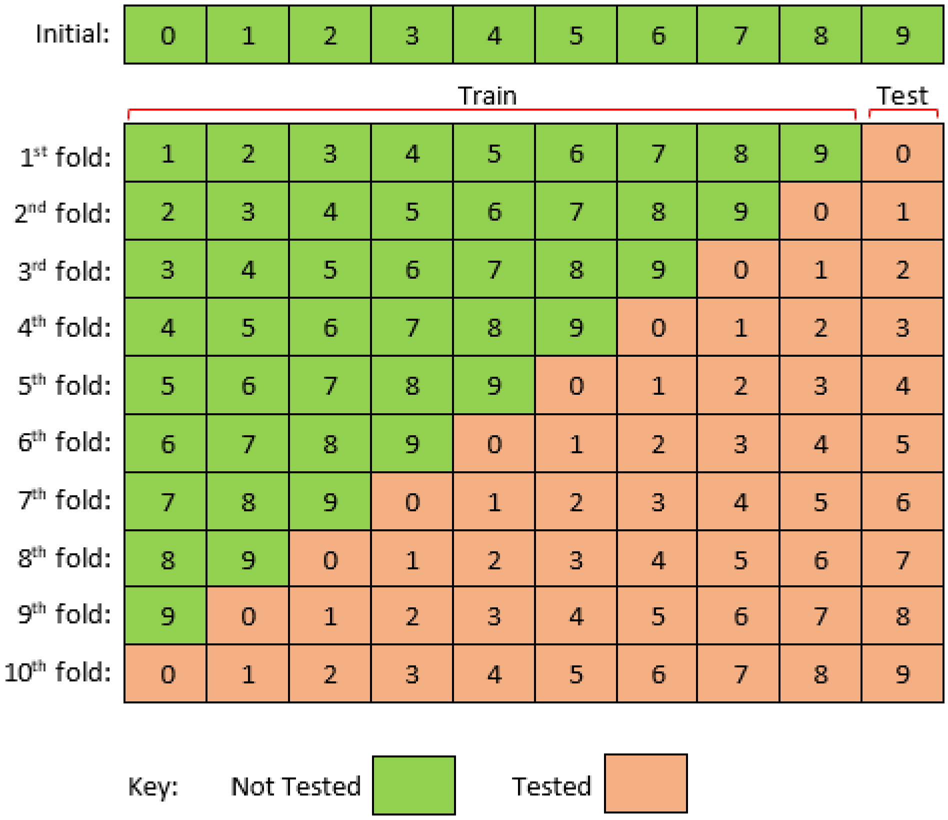
5. Set number of images to be used for training: This step depends on the type of testing procedure that is being performed, as described below.

For **CV**, the first step is to determine the size of each segment (i.e. the size of the test set for each fold). For example, if the dataset contained 100 images and we were performing 10-fold CV, each segment would contain $100/10 = 10$ images (i.e. we test 10 images each fold). In this case, `num` $= 100 - 10 = 90$ images for each fold (remember, `num` is the number of images used for training, not testing). In this example, `num` is the same for each fold; however, that is not always the case. Consider the following example, where the dataset contains 109 images and we perform 10-fold CV. Each segment would contain $109/10 = 10.9$ images. Obviously we cannot get a fraction of an image, so instead we set `num` $= 11$ images for 9-folds & `num` $= 10$ images for the remaining fold. This same principle applies whenever the dataset cannot be equally segmented (hence why we specify "roughly equally sized segments").

For **LOO**, we use only one image for testing every fold. Thus `num` = size of images-1 for every fold.

For **PI**, the CSV file should be arranged in order of subject. Then, for $fold(i)$ we have `num` = size of images – (number of images for $subject(i)$).

6. Move images & their corresponding labels for testing to end of vector: For each fold we ensure that the images to be tested are located at the end of the `rand_images` vector along with their corresponding labels in the `rand_labels` vector. We can then use the first `num` images for training and remainder for testing. The process for testing images using 10-fold CV is shown in Figure 6.2.



Initial:	0	1	2	3	4	5	6	7	8	9
1st fold:	1	2	3	4	5	6	7	8	9	0
2nd fold:	2	3	4	5	6	7	8	9	0	1
3rd fold:	3	4	5	6	7	8	9	0	1	2
4th fold:	4	5	6	7	8	9	0	1	2	3
5th fold:	5	6	7	8	9	0	1	2	3	4
6th fold:	6	7	8	9	0	1	2	3	4	5
7th fold:	7	8	9	0	1	2	3	4	5	6
8th fold:	8	9	0	1	2	3	4	5	6	7
9th fold:	9	0	1	2	3	4	5	6	7	8
10th fold:	0	1	2	3	4	5	6	7	8	9

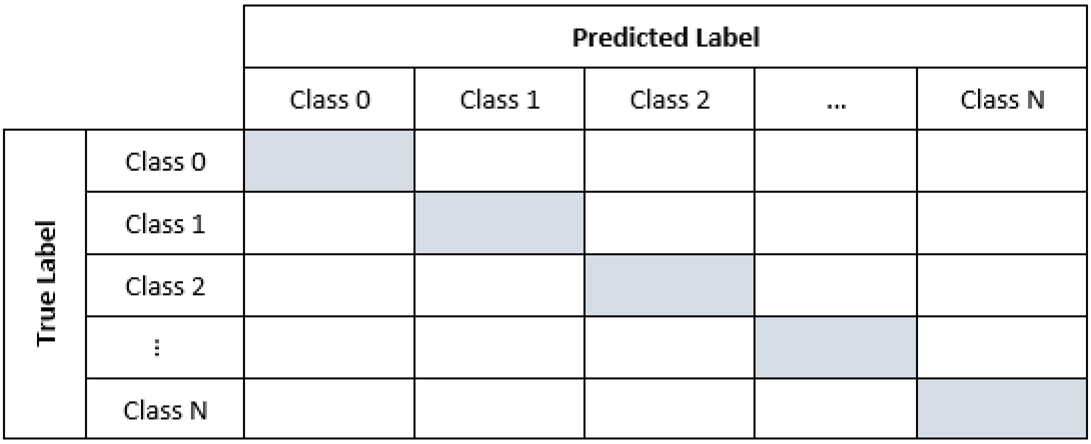
Figure 6.2: The image testing process for 10-fold cross-validation

Initially, we have a vector of images that we split into 10 roughly equally sized segments (0-9) as shown above. At this point, we have not yet tested any of the segments. For the first fold, we move the first segment, segment 0, to the end and use it as the test set. The remaining segments are used as the training set. After we complete the first fold, segment 0 will have been tested and will be included as part of the training set for all future folds. We repeat the same process for the remaining folds and see that after 10 folds have been completed, all segments will have been tested once and the images are back in the original order. We can apply the same process to LOO & PI procedures; only the size of the segments and number of folds will differ.

7. Create training data & train classifier: We perform feature extraction on each training image and pass the feature vectors to the classifier. The number of training images is indicated by `num` and because the test images are located at the end of the vector, we simply loop through all images 0 to (`num` - 1) to create the training data.

8. Create test data & predict: Now that we have trained a classifier, we perform feature extraction on each test image and predict the class label it belongs to. The test images are located at the end of the vector, from `num` to (size of images$-$ 1). We check to see if the predicted class label is correct by comparing it to the true label stored in the `rand_labels` vector. If it is correct we increment a counter, `count`, which will keep track of how many of the test images were correctly predicted. This process is repeated for each test image.

At this stage, we also increment the relevant entry of our confusion matrix. The format of the confusion matrix is shown in Figure 6.3, where each row represents the true class label and each column represents the predicted class label. Using the confusion matrix, we can determine the level of confusion between different classes of emotion.



		Predicted Label				
		Class 0	Class 1	Class 2	...	Class N
True Label	Class 0	▨				
	Class 1		▨			
	Class 2			▨		
	⋮				▨	
	Class N					▨

Figure 6.3: Confusion matrix format

9. Determine recognition accuracy for the fold: After having tested each of the images in the fold, we compute the recognition accuracy for the fold using Equation 6.2. We then add this value to the vector `percentages` to keep track of each fold's recognition accuracy.

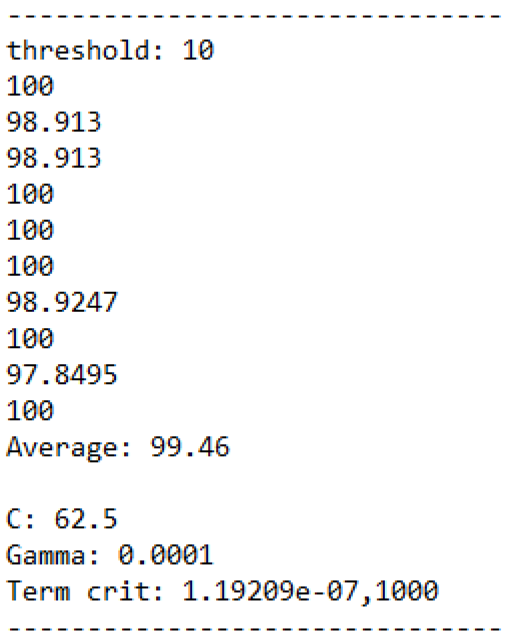
$$\text{Recognition accuracy for the fold} = \frac{count}{\text{size of images - } num} \times 100\% \qquad (6.2)$$

10. Repeat steps 5 to 9 for each remaining fold: We simply repeat the relevant steps of the testing procedure for each of the remaining folds to ensure that each segment gets tested exactly once. For 10-fold CV, we have a total of 10 folds. For LOO, the number of folds is equal to the number of images and for PI, the number of folds is equal to the number of subjects.

11. Determine average recognition accuracy across all folds: For the final step of the testing procedure, we compute the average recognition accuracy across all folds. To compute the average recognition accuracy, we sum all the percentages from each fold and divide the result by the number of folds as shown in Equation 6.3. Note that for the cross-validation procedure, we repeat steps 4 to 11 a total of 10 times. This ensures that we generate a different randomization for each run. To compute the overall average recognition accuracy across all 10 runs, we simply sum the average recognition accuracies and divide by 10.

$$\text{Average recognition accuracy} = \frac{\sum_{i=0}^{\text{number of folds}-1} percentages(i)}{\text{number of folds}} \tag{6.3}$$

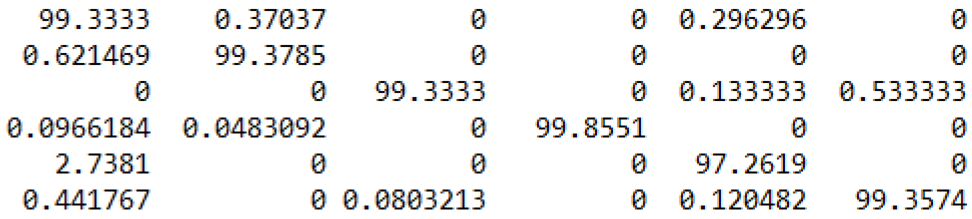
We output the average recognition accuracy as well as each fold's recognition accuracy to a file for later reference as shown in Figure 6.4.



```
-------------------------------
threshold: 10
100
98.913
98.913
100
100
100
98.9247
100
97.8495
100
Average: 99.46

C: 62.5
Gamma: 0.0001
Term crit: 1.19209e-07,1000
-------------------------------
```

Figure 6.4: Outputting the results from the test to a file

We also convert our confusion matrix to percentages by dividing each row by the number of images for that particular class and multiplying by 100%. We output the result to file as shown in Figure 6.5.



```
  99.3333   0.37037         0         0  0.296296         0
 0.621469   99.3785         0         0         0         0
        0         0   99.3333         0  0.133333  0.533333
0.0966184 0.0483092         0   99.8551         0         0
   2.7381         0         0         0   97.2619         0
 0.441767         0 0.0803213         0  0.120482   99.3574
```

Figure 6.5: Outputting the confusion matrix to a file

6.2 Datasets

This section presents the datasets that were used for testing our proposed methodology. For testing methods of Facial Expression Recognition (FER), the CK/CK+ and JAFFE datasets were used. For testing face detection and our proposed crowd emotion estimation algorithm, the previously discussed Crowd Emotion dataset was used.

6.2.1 CK/CK+

The CK/CK+ dataset is one of the most commonly used facial expression datasets in current literature. The Cohn-Kanade (CK) AU-Coded Expression dataset [52] consists of 97 university students between 18 to 30 years of age. 65% were female, 15% were African-American and 3% were Asian or Latino. Subjects were asked to perform up to 6 different prototypic emotions (i.e. joy, surprise, anger, fear, disgust and sadness) as well as a neutral expression. Image sequences from the neutral expression to target expression were captured using a frontal facing camera and digitized to 640 × 480 or 490 pixels in Portable Network Graphics (PNG) image format.

Released in 2010, The Extended Cohan-Kanade (CK+) dataset [51] increases the number of subjects from CK by 27% to 123 subjects and the number of image sequences by 22% to 593 sequences. Each peak expression is fully FACS coded and where applicable is assigned a prototypic emotion label.

In our setup, 309 sequences were selected from 106 subjects by eliminating sequences that did not belong to one of the 6 previously mentioned prototypic emotions based on the provided validated emotion labels. For 6-class expression recognition, the 3 most expressive images from each sequence were selected; resulting in 927 images. To build the 7-class dataset, the first image (neutral expression) from each of the 309 sequences was selected and added to the 6-class dataset; resulting in a total

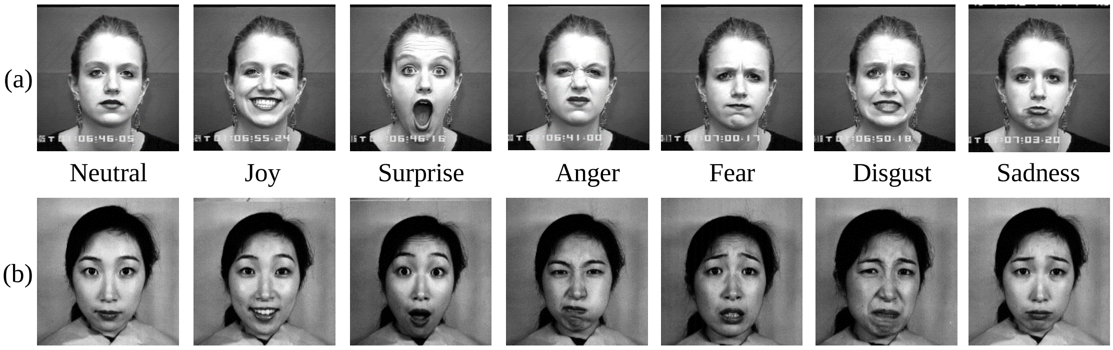


Figure 6.6: Sample prototypic expressions from (a) CK/CK+ dataset and (b) JAFFE dataset

of 1236 images. Figure 6.6a shows a sample of 7 prototypic expressions from the CK/CK+ dataset.

6.2.2 JAFFE

The Japanese Female Facial Expression (JAFFE) dataset [50] contains 213 images of 10 female Japanese subjects. Each subject was asked to perform multiple poses of 7 basic prototypic expressions. The expression label for each image represents the expression that the subject was asked to pose. The images are provided in .tiff image format with a resolution of 256×256 pixels. Figure 6.6b shows a sample of 7 prototypic expressions from the JAFFE dataset.

6.2.3 Crowd Emotion Dataset

The creation of a novel crowd dataset with known ground-truth emotion labels was discussed in Chapter 4. The use of the Crowd Emotion dataset for testing face detection and our proposed crowd emotion estimation algorithm is discussed in Sections 6.4 and 6.5 respectively.

6.3 Improved GLTP

6.3.1 Introduction

In the following sections, we analyze, discuss & compare our experimental results for GLTP [42] with our proposed method, Improved GLTP, using the CK/CK+ and JAFFE datasets. Firstly, in Section 6.3.2, we discuss the procedure for determining the optimal parameters for feature extraction and classification for both GLTP & Improved GLTP. Then, in Sections 6.3.3 & 6.3.4, we critically analyze & compare the accuracy and performance of our proposed method with GLTP. In Section 6.3.5, we compare our results to those in current literature. Finally, in Section 6.3.6, we conclude our findings.

6.3.2 Optimal Parameter Selection

In this section, we discuss the procedure that was used for finding the optimal parameters for feature extraction & classification. For GLTP, we require optimal values for the threshold (t) & region size during feature extraction, and C & γ during classification. For Improved GLTP we also need to find the optimal number of principle components for PCA. To find each optimal parameter we hold all but one parameter constant and modify the parameter of interest while performing 10-fold cross-validation on 7-classes of emotion in the CK/CK+ and JAFFE datasets. The value resulting in the highest recognition accuracy is chosen as the optimal parameter.

To begin with, we set the starting value of $t = 10$ for both GLTP & Improved GLTP as in [42]. We do not test varying region sizes for GLTP as it has already been shown that a region size of 7×6 is optimal [40][42]. For Improved GLTP we set the starting region size of each facial component as 1×1 region. We only implement & vary PCA after we have found the optimal parameters for t & the region size. To set the starting values for C & γ, we use the OpenCV [93] function *train_auto*() [94] to find good (but not necessarily optimal) starting values. More information about finding each optimal parameter is given below and the results for the CK/CK+ dataset are summarized in Table 6.1 (for GLTP) and Table 6.2 (for Improved GLTP).

Threshold

Starting with $t = 10$ we consecutively increment t by 10 up until a threshold value of $t = 50$. We then refine the value around the threshold that had the highest recognition accuracy. Thresholds of $t = 10$ and $t = 20$ were found to be the optimal values for the CK/CK+ and JAFFE datasets respectively for both GLTP & Improved GLTP.

Region size

GLTP: Before extracting GLTP histograms from the face, it is first segmented into $m \times n$ regions. This ensures that location information is included together with frequency occurrence information. Having few regions improves classification performance, however may result in a low recognition accuracy. On the other hand, having too many regions reduces classification efficiency and may also lower recognition accuracy due to too much unnecessary location information being included. It has been shown in literature that a region size of 7×6 results in optimal recognition accuracy and efficiency [40][42].

Improved GLTP: The technique of multiple region segmentation is also used when working with facial components. Region sizes of 1×1, 2×2, 2×4, 3×4 and 3×5 were tested in succession. A region size of 3×4 resulted in optimal recognition accuracy.

(Only for Improved GLTP) Principle Component Analysis (PCA)

In PCA, the number of components selected for projection is a trade-off between computational efficiency and recognition accuracy. If too few features remain after applying PCA, efficiency will be high but recognition accuracy will decrease as not enough discriminative features remain in the feature vector. On the other hand, having too large a number of features remaining will not result in any improvement to efficiency. Principle component values of 64, 128, 256, 512 and 1024 were tested and it was found that the number of principle components that resulted in optimal accuracy and efficiency was 256 components for the CK/CK+ dataset and 64 components for the JAFFE dataset. Also see Section 6.3.4, where we investigated the effect of PCA on system efficiency.

Classification using Support Vector Machines (SVM)

An optimal parameter grid-search was carried out to find the values C and γ for the SVM as outlined in [46]. We found that varying the C value past $C = 10$ had no effect on recognition accuracy. Thus we left the optimal C value as the value returned by the *train_auto*() function (which was within that range). After identifying a good value for γ from the initial grid-search, we further refined the grid around that value to find the optimal value.

Table 6.1: Optimal parameter selection for GLTP using CK/CK+ dataset

t	C	γ	Recognition Accuracy (%)
1. Find optimal threshold			
10	12.5	1.5e-4	96.5
20	12.5	1.5e-4	96.1
30	12.5	1.5e-4	95.2
40	12.5	1.5e-4	94.1
50	12.5	1.5e-4	89.0
1.1 Refine threshold: $(0 < t < 20)$			
5	12.5	1.5e-4	96.2
10	12.5	1.5e-4	96.5
15	12.5	1.5e-4	96.3
2. Find optimal C value			
10	0.1	1.5e-4	63.3
10	1	1.5e-4	91.7
10	10	1.5e-4	96.5
10	12.5	1.5e-4	96.5
10	100	1.5e-4	96.5
10	1000	1.5e-4	96.5
3. Find optimal γ value			
10	12.5	1e-5	92.3
10	12.5	2.5e-5	93.9
10	12.5	5e-5	94.4
10	12.5	7.5e-5	95.7
10	12.5	1e-4	96.9
10	12.5	2.5e-4	95.5
10	12.5	5e-4	94.2
3.1 Refine γ: (7.5e-5 $< \gamma <$ 2.5e-4)			
10	12.5	0.5e-4	96.6
10	12.5	1e-4	96.9
10	12.5	1.5e-4	96.5
4. Optimal parameters are:			
10	12.5	1e-4	96.9

Table 6.2: Optimal parameter selection for Improved GLTP using CK/CK+ dataset

t	Region size	PCA components	C	γ	Recognition Accuracy (%)
1. Find optimal threshold					
10	1×1	-	62.5	1e-5	76.2
20	1×1	-	62.5	1e-5	75.1
30	1×1	-	62.5	1e-5	73.7
40	1×1	-	62.5	1e-5	68.9
50	1×1	-	62.5	1e-5	63.8
1.1 Refine threshold: $(0 < t < 20)$					
5	1×1	-	62.5	1e-5	75.7
10	1×1	-	62.5	1e-5	76.2
15	1×1	-	62.5	1e-5	75.5
2. Find optimal region size					
10	1×1	-	62.5	1e-5	76.2
10	2×2	-	62.5	1e-5	92.1
10	2×4	-	62.5	1e-5	95.1
10	3×4	-	62.5	1e-5	96.0
10	3×5	-	62.5	1e-5	95.7
3. Find optimal number of PCA components					
10	3×4	64	62.5	1e-5	96.4
10	3×4	128	62.5	1e-5	96.8
10	3×4	256	62.5	1e-5	97.1
10	3×4	512	62.5	1e-5	96.7
10	3×4	1024	62.5	1e-5	96.5
4. Find optimal C value					
10	3×4	256	0.1	1e-5	72.3
10	3×4	256	1	1e-5	92.4
10	3×4	256	10	1e-5	97.1
10	3×4	256	62.5	1e-5	97.1
10	3×4	256	100	1e-5	97.1
10	3×4	256	1000	1e-5	97.1
5. Find optimal γ value					
10	3×4	256	62.5	1e-5	97.1
10	3×4	256	62.5	2.5e-5	97.3
10	3×4	256	62.5	5e-5	97.6
10	3×4	256	62.5	7.5e-5	97.2
10	3×4	256	62.5	1e-4	96.9
10	3×4	256	62.5	2.5e-4	96.2
10	3×4	256	62.5	5e-4	95.6
5.1 Refine γ: (2.5e-5 $< \gamma <$ 7.5e-5)					
10	3×4	256	62.5	4e-5	97.4
10	3×4	256	62.5	5e-5	97.6
10	3×4	256	62.5	6e-5	97.3
6. Optimal parameters are:					
10	3×4	256	62.5	5e-5	97.6

6.3.3 Recognition accuracy

We test Improved GLTP using 10-fold cross-validation, leave-one-out & person independent testing procedures (see Section 6.1) for 6 & 7 classes of emotion in the CK/CK+ dataset (see Section 6.2). Tables 6.3 & 6.4 summarize the recognition accuracies obtained for GLTP & our proposed method, Improved GLTP, when using optimal parameters.

Table 6.3: Recognition accuracy (%) for 6-class emotion in the CK/CK+ dataset using GLTP and Improved GLTP

Method	**Cross-validation**	**Leave-one-out**	**Person Independent**
GLTP	98.9 ± 0.16	99.2	86.4
Improved GLTP	99.3 ± 0.25	99.7	86.5

Table 6.4: Recognition accuracy (%) for 7-class emotion in the CK/CK+ dataset using GLTP and Improved GLTP

Method	**Cross-validation**	**Leave-one-out**	**Person Independent**
GLTP	96.9 ± 0.21	97.4	83.0
Improved GLTP	97.6 ± 0.30	98.1	83.1

We observe that our proposed method outperforms traditional GLTP by 0.4% to 0.7% in cross-validation testing and 0.5% to 0.7% in leave-one-out testing for 6 and 7 classes of emotion respectively. The results for person independent testing also shows a slight improvement when using Improved GLTP. The results confirm that the proposed enhancements to GLTP such as the use of the more accurate Scharr gradient operator, dimensionality reduction via PCA, and facial component extraction, when combined, further improves the recognition accuracy of GLTP.

Table 6.5: Confusion matrix (%) of cross-validation testing for 6-classes of emotion in the CK/CK+ dataset when using Improved GLTP

	Joy	**Sur**	**Ang**	**Fear**	**Dis**	**Sad**
Joy	**99.9**	0	0.1	0	0	0
Sur	0	**99.4**	0.4	0.1	0	0.1
Ang	0	0	**99.3**	0	0.4	0.3
Fear	0	0.5	0	**99.4**	0	0.1
Dis	0	0	0.6	0	**99.4**	0
Sad	0	0	2.7	0	0	**97.3**

Table 6.6: Confusion matrix (%) of cross-validation testing for 7-classes of emotion in the CK/CK+ dataset when using Improved GLTP

	Joy	**Sur**	**Ang**	**Fear**	**Dis**	**Sad**	**Neu**
Joy	**100**	0	0	0	0	0	0
Sur	0.5	**98**	0	0	0	0	1.5
Ang	0	0.1	**95.3**	0	0	0	4.6
Fear	0	0.4	0	**99.2**	0	0	0.4
Dis	0	0	0.5	0	**98.6**	0	0.9
Sad	0	0	0.9	0.4	0	**93.9**	4.8
Neu	0.1	0	1.8	0.1	0.5	0.9	**96.6**

Tables 6.5 and 6.6 show the confusion matrices of 10-fold cross validation testing for 6 and 7 classes of emotion in the CK/CK+ dataset when using Improved GLTP. We observe that, in particular, anger and sadness emotions get confused with neutral emotion in 7 class cross-validation testing.

Table 6.7: Confusion matrix (%) of leave-one-out testing for 6-classes of emotion in the CK/CK+ dataset using Improved GLTP

	Joy	**Sur**	**Ang**	**Fear**	**Dis**	**Sad**
Joy	**100**	0	0	0	0	0
Sur	0	**99.6**	0.4	0	0	0
Ang	0	0	**100**	0	0.4	0.3
Fear	0	0.5	0	**100**	0	0.1
Dis	0	0	0.6	0	**100**	0
Sad	0	0	2.4	0	0	**97.6**

Table 6.8: Confusion matrix (%) of leave-one-out testing for 7-classes of emotion in the CK/CK+ dataset using Improved GLTP

	Joy	**Sur**	**Ang**	**Fear**	**Dis**	**Sad**	**Neu**
Joy	**100**	0	0	0	0	0	0
Sur	0.4	**98.4**	0	0	0	0	1.2
Ang	0	0.1	**97**	0	0	0	3
Fear	0	0	0	**100**	0	0	0
Dis	0	0	0.5	0	**99.4**	0	0.6
Sad	0	0	1.2	0	0	**95.2**	3.6
Neu	0	0	1.6	0	0.6	1	**96.8**

Tables 6.7 and 6.8 show the confusion matrices of leave-one-out testing for 6 and 7 classes of emotion in the CK/CK+ dataset when using Improved GLTP. As expected, the leave-out-out testing procedure achieved the highest recognition accuracies on test. This is because, for each fold, the entire database apart from one image is used as the training set. Although high accuracies were achieved for both 6 and 7 classes of emotion, we observe that, anger and sadness emotions do get confused with the added neutral emotion in 7 class testing; reducing the overall recognition accuracy slightly.

Table 6.9: Confusion matrix (%) of person independent testing for 6-classes of emotion in CK/CK+ dataset using Improved GLTP

	Joy	**Sur**	**Ang**	**Fear**	**Dis**	**Sad**
Joy	**96.6**	0	1.9	0	1.5	0
Sur	0.8	**97.2**	1.2	0.4	0.4	0
Ang	0	4.4	**78.5**	0	11.9	5.2
Fear	16	6.7	6.7	**58.6**	0	12
Dis	2.3	0	11.3	0	**86.4**	0
Sad	0	11.9	14.3	8.3	0	**65.5**

Table 6.10: Confusion matrix (%) of person independent testing for 7-classes of emotion in CK/CK+ dataset using Improved GLTP

	Joy	**Sur**	**Ang**	**Fear**	**Dis**	**Sad**	**Neu**
Joy	**93.7**	0	0.5	0	1.0	0	4.8
Sur	0.8	**94**	0	0.4	0	0	4.8
Ang	0	1.5	**61.5**	0	8.1	0	28.9
Fear	13.3	2.7	4	**56**	0	4	20
Dis	2.3	0	6.2	0	**82.5**	0	9
Sad	0	3.6	3.6	6	0	**48.8**	38
Neu	0.3	1	2.5	1	1.3	1.3	**92.6**

Tables 6.9 and 6.10 show the confusion matrices of person independent testing for 6 and 7 classes of emotion in the CK/CK+ dataset using Improved GLTP. This testing procedure achieved the lowest recognition accuracies on test. This is because not a single image from the test subject remains in the training set. Person independent testing is thus used to show the robustness of the method under test to unseen subjects. We observe that in 6 class testing; anger, fear, and sadness emotions are misclassified the most. In 7 class testing, the accuracies further decrease as the emotions also get confused to a great extent with the added neutral emotion.

We repeat the cross-validation and leave-one-out tests from above on the JAFFE dataset. The results are summarized in Tables 6.11 and 6.12. We see that our proposed method, Improved GLTP, shows a significant improvement in recognition accuracy over traditional GLTP for the JAFFE dataset. The largest improvement in recognition accuracy was seen during 7-class cross-validation testing, where Improved GLTP outperformed GLTP by 7.3%. Improved GLTP also showed a sizeable improvement of 7%, 6.3% and 5.5% for 7-class leave-one-out, 6-class cross-validation and 6-class leave-one-out tests respectively. The results from the JAFFE dataset further verifies that the recognition accuracy of Improved GLTP is superior to that of traditional GLTP.

Table 6.11: Recognition accuracy (%) for 6-class emotion in the JAFFE dataset using GLTP and Improved GLTP

Method	**Cross-validation**	**Leave-one-out**
GLTP	77.0 ± 1.1	81.3
Improved GLTP	83.3 ± 1.6	86.8

Table 6.12: Recognition accuracy (%) for 7-class emotion in the JAFFE dataset using GLTP and Improved GLTP

Method	**Cross-validation**	**Leave-one-out**
GLTP	74.4 ± 1.3	77.5
Improved GLTP	81.7 ± 1.3	84.5

The confusion matrices for cross-validation and leave-one-out tests on the JAFFE dataset are provided in Tables 6.13 to 6.16. We observe that, like with the CK/CK+ dataset, leave-one-out testing outperformed cross-validation testing while 7-class testing produced lower recognition accuracies compared to 6-class testing due to the added confusion caused by the included neutral emotion. We also observe that the disgust emotion is the only emotion to not be confused with neutral emotion in 7-class testing.

Table 6.13: Confusion matrix (%) of cross-validation testing for 6-classes of emotion in the JAFFE dataset when using Improved GLTP

	Joy	**Sur**	**Ang**	**Fear**	**Dis**	**Sad**
Joy	**89.1**	0.3	0	3.1	0.6	6.9
Sur	3.4	**84.5**	4.1	5.2	0.7	2.1
Ang	0	1.3	**81.7**	6	9.3	1.7
Fear	3.1	7.5	0.9	**75.4**	5	8.1
Dis	0	0	5.5	1.4	**89**	4.1
Sad	0	5.3	0	7	7.3	**80.4**

Table 6.14: Confusion matrix (%) of cross-validation testing for 7-classes of emotion in the JAFFE dataset when using Improved GLTP

	Joy	**Sur**	**Ang**	**Fear**	**Dis**	**Sad**	**Neu**
Joy	**86.9**	0	0	3.4	0	0.3	9.4
Sur	0	**76.6**	3.1	1.7	0.7	0	17.9
Ang	0	2.3	**78.4**	4.7	10.3	0.3	4
Fear	1.3	5.9	0.3	**74.1**	4.1	6.9	7.4
Dis	0	0	4.1	1.4	**90.4**	4.1	0
Sad	0	5	1	8.6	8	**73.7**	3.7
Neu	0	3.2	4.5	0	0	0.3	**92**

Table 6.15: Confusion matrix (%) of leave-one-out testing for 6-classes of emotion in the JAFFE dataset using Improved GLTP

	Joy	**Sur**	**Ang**	**Fear**	**Dis**	**Sad**
Joy	**90.6**	0	0	3.1	0	6.3
Sur	3.4	**89.8**	3.4	3.4	0	0
Ang	0	0	**86.7**	3.3	10	0
Fear	0	9.3	0	**78.1**	6.3	6.3
Dis	0	0	3.4	0	**93.2**	3.4
Sad	0	6.7	0	3.3	6.7	**83.3**

Table 6.16: Confusion matrix (%) of leave-one-out testing for 7-classes of emotion in the JAFFE dataset using Improved GLTP

	Joy	**Sur**	**Ang**	**Fear**	**Dis**	**Sad**	**Neu**
Joy	**87.5**	0	0	3.1	0	0	9.4
Sur	0	**79.4**	3.4	0	0	0	17.2
Ang	0	0	**83.4**	3.3	10	0	3.3
Fear	0	6.3	0	**75**	6.2	6.3	6.2
Dis	0	0	3.4	0	**93.2**	3.4	0
Sad	0	6.7	0	3.3	6.7	**80**	3.3
Neu	0	3.2	3.2	0	0	0	**93.6**

6.3.4 Effect of PCA on performance

In Section 6.3.2, we found that for the CK/CK+ dataset, the highest recognition accuracy was achieved with 256 principle components selected for projection. However, for the value to be optimal, we also consider the effect it has on performance. If, for example, too many components are set to remain, the time taken to project the increased number of components will outweigh the efficiency improvement gained from training/testing the reduced feature set and efficiency will instead decrease.

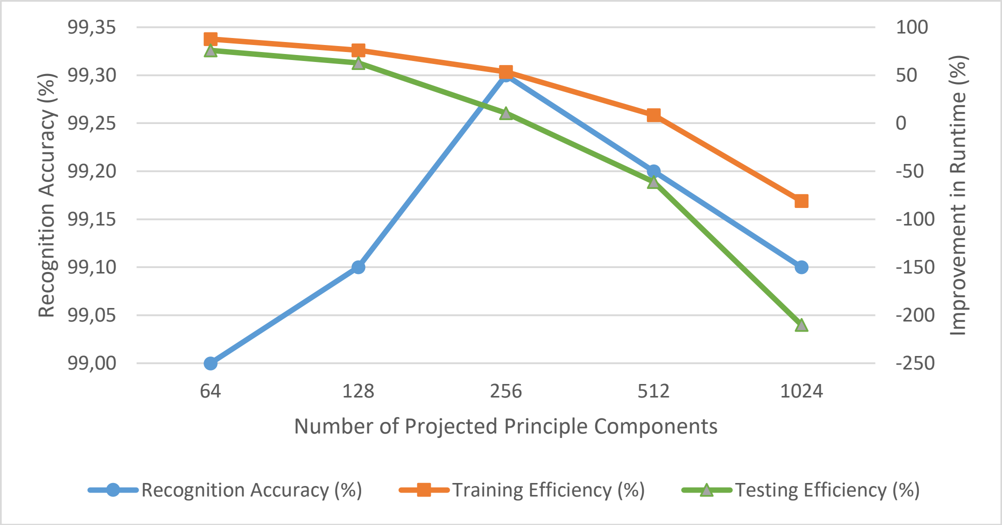


Figure 6.7: The effect of varying the number of projected principle components on recognition accuracy and runtime for 6 classes of emotion in the CK/CK+ dataset

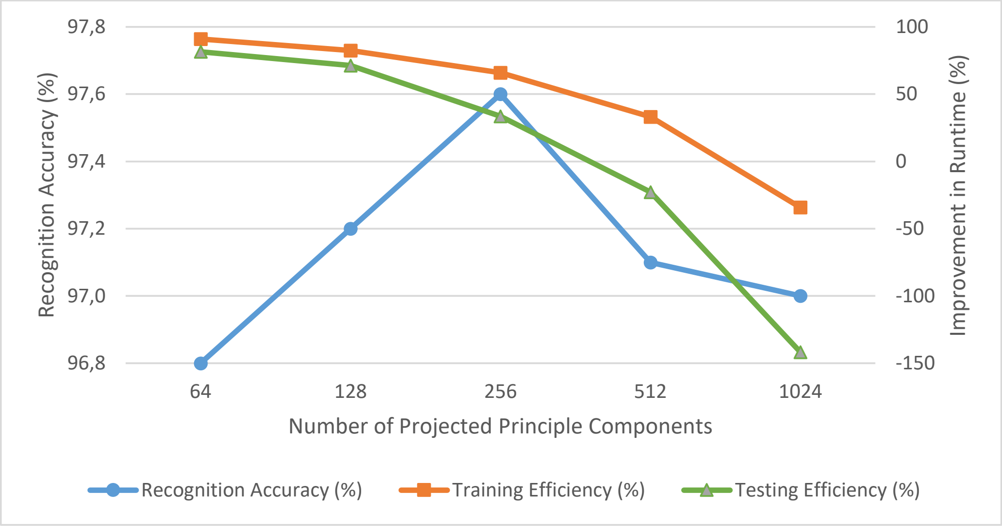
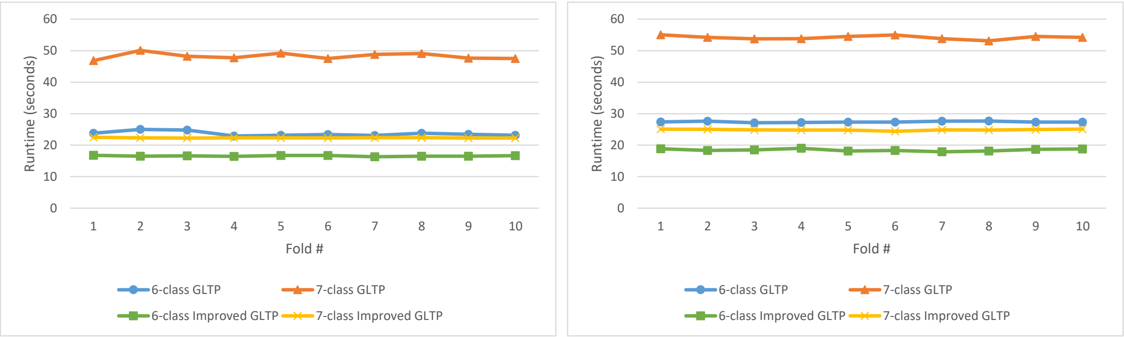


Figure 6.8: The effect of varying the number of projected principle components on recognition accuracy and runtime for 7 classes of emotion in the CK/CK+ dataset

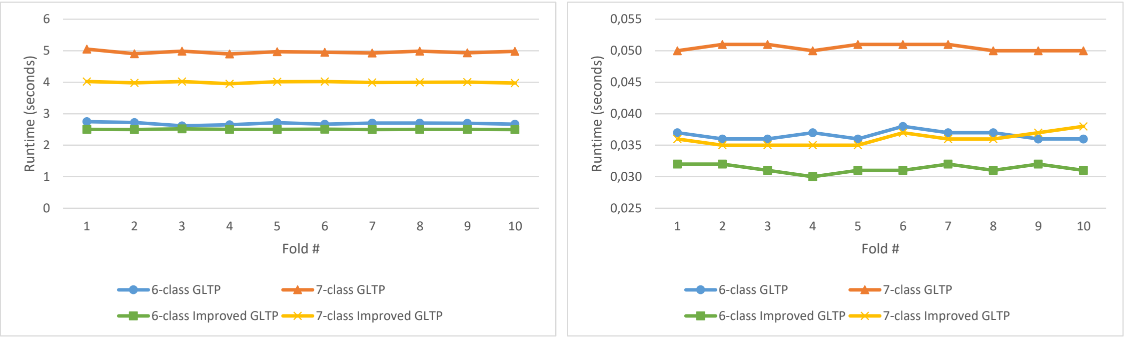
Figures 6.7 and 6.8 demonstrate the effect of varying the number of principle components on recognition accuracy and training/testing runtime using 10-fold cross-validation for 6 and 7 classes of emotion in the CK/CK+ dataset. The improvement in runtime is in relation to running Improved GLTP without PCA. The results show that projecting a small number of principle components, such as 64, results in a large decrease in runtime but a comparatively lower recognition accuracy. On the other hand, projecting a larger number of principle components, such as 1024, actually increases runtime while recognition accuracy decreases as a result of redundant data being included. It can be seen that the optimal number of principle components to be used for projection is indeed 256.



(a) Cross-validation

(b) Leave-one-out

Figure 6.9: Training Runtime



(a) Cross-validation

(b) Leave-one-out

Figure 6.10: Testing Runtime

We compare the performance of GLTP to Improved GLTP by comparing the runtime of the training and testing stages for 10-fold cross-validation & leave-one-out procedures. We do not consider the person independent procedure as the size of the training & test segments varies per fold. Figures 6.9 & 6.10 show the training & testing runtime for the first 10 folds using the CK/CK+ dataset (Core 2 Duo, 2.0GHz, 3GB RAM). The average per fold runtime for each method is given in Table 6.17.

Table 6.17: Average Runtime per fold (seconds) using the CK/CK+ dataset

(a) GLTP

	Training		**Testing**	
	6-class	7-class	6-class	7-class
CV	23.65	48.27	2.69	4.96
LOO	27.39	54.20	0.037	0.051

(b) Improved GLTP

	Training		**Testing**	
	6-class	7-class	6-class	7-class
CV	16.58	22.33	2.50	3.99
LOO	18.46	24.86	0.031	0.036

The improvement in runtime when using Improved GLTP is shown in Table 6.18.

Table 6.18: Improvement in Runtime when using Improved GLTP on CK/CK+ dataset

(a) Per fold improvement (seconds)

	Training		**Testing**	
	6-class	7-class	6-class	7-class
CV	7.07	25.94	0.19	0.97
LOO	8.93	29.34	0.006	0.015

(b) Percentage improvement (%)

	Training		**Testing**	
	6-class	7-class	6-class	7-class
CV	29.9	53.7	7.1	19.6
LOO	32.6	54.1	16.2	29.4

The overall improvement in runtime across all folds is shown in Table 6.19. For 10-fold cross-validation, we simply multiply the per fold improvement by 10. For leave-one-out, we multiply the per fold improvement by 927 for 6 classes of emotion & 1236 for 7 classes of emotion.

Table 6.19: Overall Improvement in Runtime when using Improved GLTP on CK/CK+ dataset

(a) (seconds)

	Training		**Testing**	
	6-class	7-class	6-class	7-class
CV	70.7	259.4	1.9	9.7
LOO	8278.1	36264.2	5.6	18.5

(b) (hours)

	Training		**Testing**	
	6-class	7-class	6-class	7-class
CV	0.0196	0.0721	0.0005	0.0027
LOO	2.30	10.07	0.0016	0.0051

The results show that dimensionality reduction via PCA had a positive effect on runtime performance. The largest improvement in runtime was seen during the training stages. In particular, for the leave-one-out procedure, Improved GLTP saved a total of 10.07 hours for 7-class training; reducing the overall training runtime by 54.1% compared to GLTP.

Table 6.20: Overall Improvement in Runtime when using Improved GLTP on JAFFE dataset

(a) (seconds)

	Training		**Testing**	
	6-class	7-class	6-class	7-class
CV	11.4	17.6	0.4	0.7
LOO	262.6	535.3	0.7	0.6

(b) Percentage (%)

	Training		**Testing**	
	6-class	7-class	6-class	7-class
CV	60.0	66.8	24.2	33.0
LOO	63.5	72.0	30.8	23.1

We also compare the performance of Improved GLTP to GLTP using the JAFFE dataset. The overall improvement in runtime across all folds when using Improved GLTP is given in Table 6.20. We confirm that Improved GLTP showed an improvement in runtime across all testing procedures. Once again, the largest improvement was seen during the training stages. From the results obtained using both datasets, we have verified that Improved GLTP is more efficient than traditional GLTP.

6.3.5 Comparison to results in literature

Table 6.21 shows a comparison between some common and state of the art methods of FER in current literature using the CK/CK+ dataset. To make a fair and accurate comparison; the results for LBP, LDP, LTP and GLTP are taken from [42], where the experimental setup and testing procedure were kept constant for all methods. In [42], a 10-fold cross-validation testing procedure was used for 6 and 7 classes of emotion in the CK/CK+ [51][52] dataset. The results confirm that LBP was the least accurate method on test. LDP and LTP outperformed LBP, achieving very similar results to one another with only a 0.1% (6 class) to 0.5% (7 class) difference in recognition accuracy. GLTP was the best performing appearance based method on test outperforming LDP by 3.3% (7 class) to 3.5% (6 class) and LTP by 2.8% (7 class) to 3.6% (6 class).

In our testing, we achieved a 98.9% and 96.9% recognition accuracy for 6 and 7 classes of emotion respectively when using GLTP with the same testing procedure. The increase in base method results by 1.7% and 5.2% respectively is possibly due to two different reasons. Firstly, our experimental setup differs to that in [42] by the fact that we selected only the images in the CK/CK+ dataset that exhibited a definitive prototypic emotion. Our selection of total images is thus 24% less than in [42]. However, we included 11% more subjects from the extended CK+ database to increase variety compared to [42]. Secondly, we refined the pre-processing of the images by further reducing the area of the cropped face. This eliminated much of the redundant areas that remained after pre-processing in [42]. The combination of including only images with definitive prototypic emotion and a refined method of pre-processing are the likely causes of the increase in the base recognition accuracy of GLTP.

We also compare our results to other state of the art methods of FER in current literature. In [65], a Deep Belief Network (DBN) was used for feature extraction and classification. A recognition rate of 91.1% was achieved for 7-classes of emotion using a 7-fold cross validation testing scheme. Then in [70], the method was improved by using a Boosted DBN (BDBN). Using this method, a recognition accuracy of 96.7% was achieved for 6-classes of emotion using an 8-fold cross-validation testing scheme. If we were to disregard any differences due to the fold size; the results are 6.5% and 2.6% less accurate than our equivalent method. The computational cost of Deep Learning methods is also much greater than Improved GLTP.

In [71], a state of the art method of dynamic FER using Atlas Construction and Sparse Representation was presented. A recognition accuracy of 97.2% was achieved for 7-classes of emotion using a person independent testing scheme. In our tests we obtained a 83.1% recognition accuracy when using Improved GLTP with the same testing procedure. The results from the two methods differ by 14.1% when temporal information is available. However, in [71], the same test was also performed without any temporal information, i.e. only spatial information was used for features selection. In this case, a recognition accuracy of 92.4% was achieved; 4.8% lower than when temporal information was included. Even without temporal information, the method still achieves a 9.3% higher recognition accuracy than that of Improved GLTP. The results show that dynamic FER can offer much higher recognition accuracies than traditional static methods of FER. However, the computational cost & complexity of working with dynamic image sequences is much greater than when working with static images. The authors report that it takes 1.6 seconds to predict one image sequence for one subject (4-core, 2.5GHz, 6GB RAM). In comparison, we found that our proposed method, Improved GLTP, took an average time of under 40 milliseconds for each image prediction (2-Core, 2.0GHz, 3GB RAM). So although the method is very accurate, it is unfeasible for our required application of real-time Crowd Monitoring.

6.3.6 Section Summary

In this section, we confirmed, via extensive testing on the CK/CK+ and JAFFE datasets, that Improved GLTP is an accurate and efficient feature extraction technique well suited for the task of FER. We found that Improved GLTP outperformed GLTP [42] in terms of both recognition accuracy & efficiency. In a comparison with other state of the art methods in current literature, we found that Improved GLTP offers an optimal solution to FER in terms of its robustness, high recognition accuracy & real-time performance characteristics.

Table 6.21: Comparison of recognition accuracies (%) for various methods of FER using the CK/CK+ dataset

Method$^{(1)}$	**6-class**	**7-class**
LBP [42]	90.1	83.3
LDP [42]	93.7	88.4
LTP [42]	93.6	88.9
GLTP [42]	97.2	91.7
DBN [65]	-	91.1$^{(2)}$
BDBN [70]	96.7$^{(3)}$	-
Dynamic Atlas [71]	-	97.2$^{(4)}$ 92.4$^{(4)(5)}$
Improved GLTP	**99.3**	**97.6**

$^{(1)}$10-fold cross-validation unless otherwise stated

$^{(2)}$7-fold cross-validation

$^{(3)}$8-fold cross-validation

$^{(4)}$person independent

$^{(5)}$without temporal information

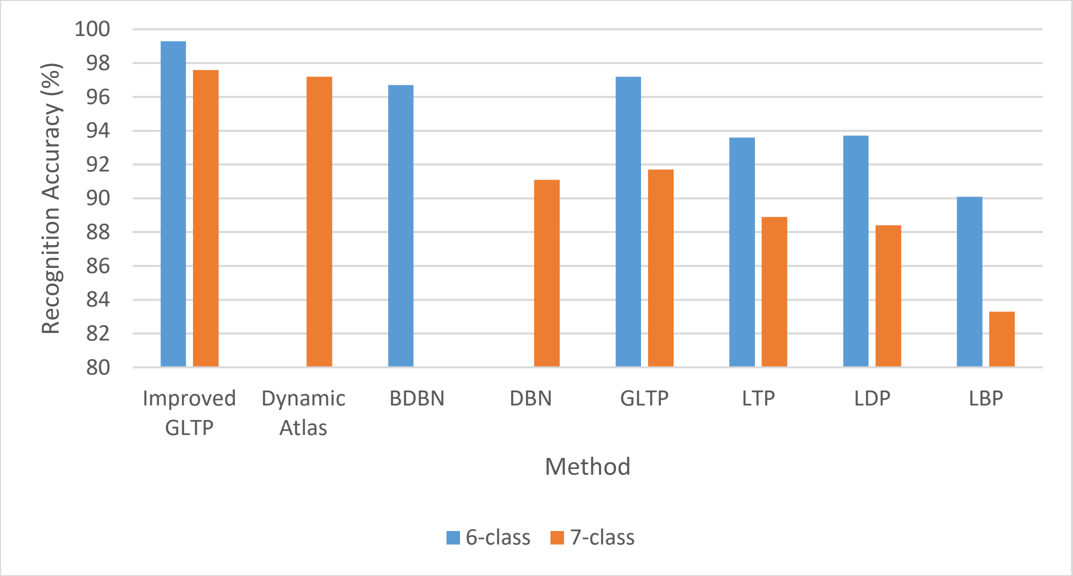


Figure 6.11: Comparison of some methods of FER for 6 & 7 classes of emotion in CK/CK+ dataset

6.4 Face Detection

The Viola and Jones face detection algorithm [33] is included as part of the OpenCV image processing library [93]. Two parameters must be correctly configured for successful face detection; namely the scale factor & the minimum number of neighbouring rectangles. The Viola and Jones model is designed to detect faces of the same size as those specified during training. In order to detect faces of all sizes, the input image is scaled down to different levels; increasing the chance of finding matching sized faces. The scale factor determines the percentage by which the input image is reduced each time. To avoid false detections as well as multiple detections of the same face, we also specify the number of neighbouring rectangles that should exist after all layers have been processed in order for the detected object to pass as a face. Objects that are not detected multiple times, and thus do not have the required number of neighbouring rectangles, are excluded from the detected faces.

To find the optimal values for the scale factor and minimum neighbours, we used a crowd image from our Crowd Emotion dataset as shown in Figure 4.10. The image contains 48 unobscured faces of neutral expression. We implemented the Viola and Jones face detection algorithm on the image and varied the number of minimum neighbouring rectangles while using a default scale factor of 1.1. A neighbour value of **5** resulted in all 48 faces being detected with only a few false positives and was selected as the optimal value. To further refine the detection results we adjust the value of the scale factor and investigate its effect on performance and accuracy as shown in Figure 6.12. We define performance as the time taken to perform face detection on the whole image and the accuracy as:

$$\text{Accuracy} = \frac{\text{true detections} - \text{false or repeated detections}}{\text{total faces}} \times 100\% \tag{6.4}$$

From the results we see that as the scale factor is increased, runtime decreases. This is because the larger the percentage reduction in scale is, the less layers will have

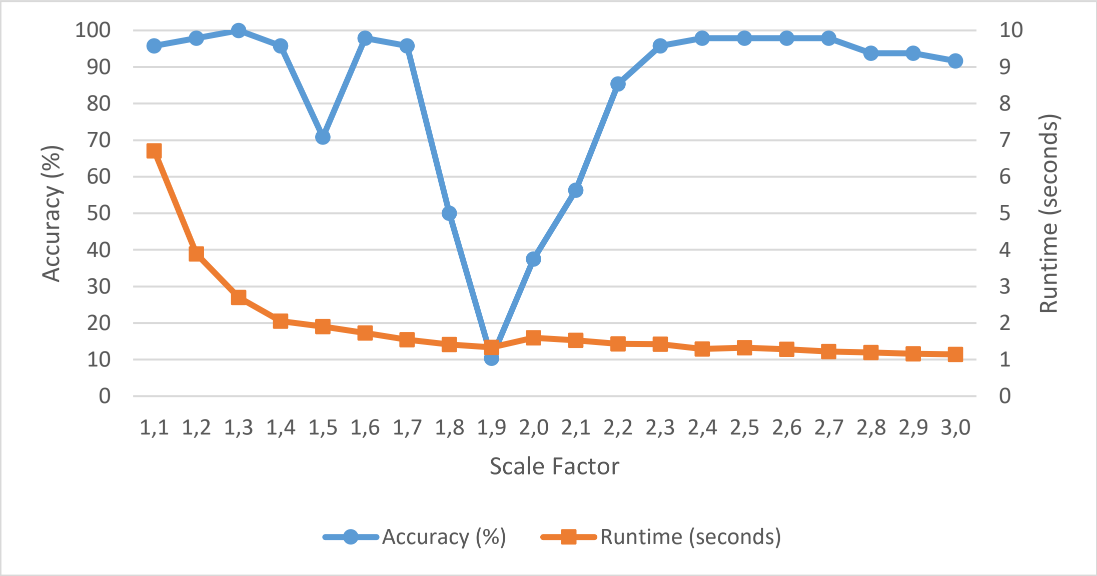


Figure 6.12: Effect of varying scale factor on accuracy & performance of face detection

to be processed. However, this also means that the number of detected faces will decrease because faces of certain sizes will be skipped due to the large jump in scale size. From Figure 6.12, we see that the optimal value for the scale factor, having achieved 100% detection accuracy in 2.7 seconds, is **1.3**. An example of incorrectly setting the parameters for scale factor and minimum neighbours is shown in Figure 6.13; many false and repeat detections are present. On the other hand, setting the parameters to their optimal values yields a high detection accuracy as shown in Figure 6.14.

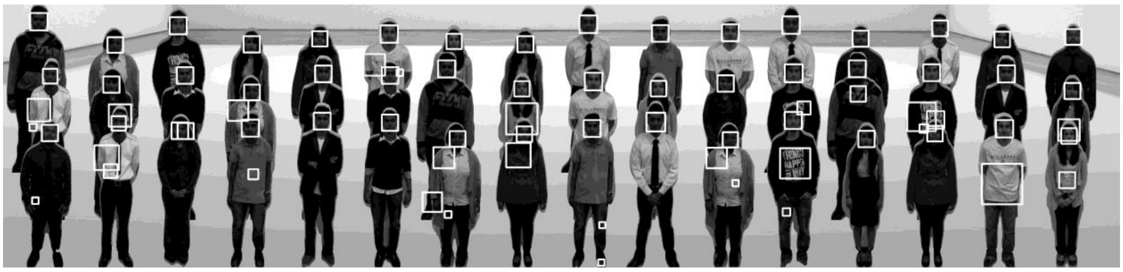


Figure 6.13: Poor face detection accuracy when using incorrect parameters

To determine the average detection accuracy for the dataset we created 100 test images, each of which consists of 24 randomly placed subjects expressing random emotions as shown in Figure 6.15. No facial occlusions are present in the images.

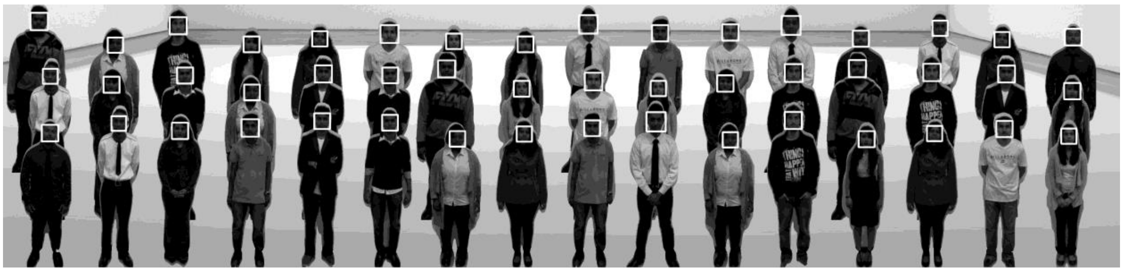


Figure 6.14: High face detection accuracy when using optimal parameters

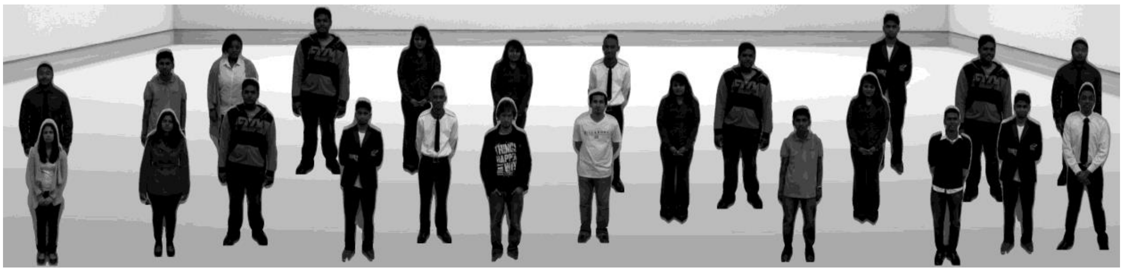


Figure 6.15: 24 randomly placed subjects expressing various emotions

Face detection using the Viola and Jones algorithm with optimal parameters was performed on each image. The detection accuracy for each image was calculated using Equation 6.4 and the results are summarized in Figure 6.16. The results show that the Viola and Jones face detection algorithm performs very well on our dataset; achieving over 83% detection accuracy for all images tested. The average detection accuracy across all 100 images on test was found to be **95.9%**, which serves as the current benchmark for our Crowd Emotion dataset.

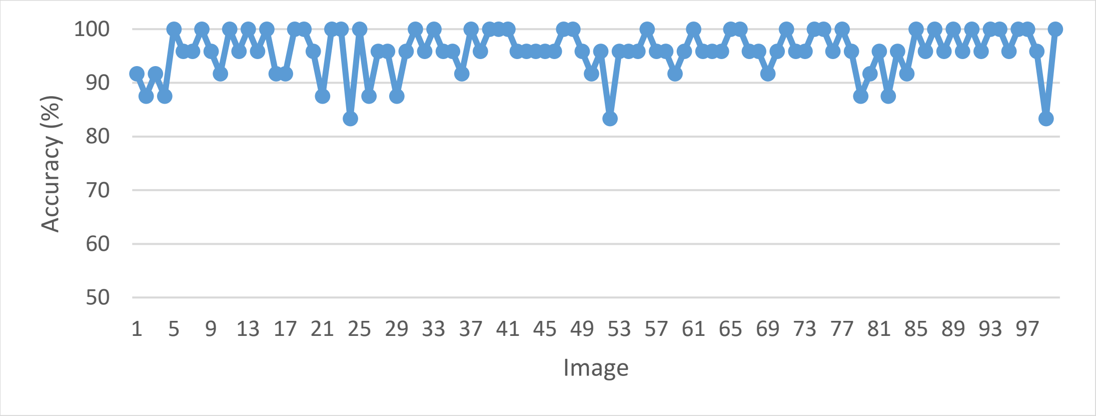


Figure 6.16: Accuracy of face detection across 100 crowd images

6.5 Predicting the Emotion of a Crowd

6.5.1 Introduction

In the following sections, the experimental results for our proposed crowd emotion estimation algorithm are presented, analyzed and discussed in detail. In Section 6.5.2, we discuss the procedure for determining the accuracy of our proposed method and the results are presented. In Section 6.5.3, we analyze and discuss the performance of our proposed method. We conclude our findings in Section 6.5.4.

6.5.2 Recognition Accuracy

To find the average recognition accuracy of our proposed crowd emotion estimation algorithm, we implement a 10-fold cross-validation testing procedure using the subject images captured in Chapter 4. The subject images and their corresponding masks are randomized and divided into 10 roughly equally-sized segments. For each fold, 9 of the segments are used for training the classifier (see Section 3.4) while the remaining segment is used to generate crowd images for testing (see Chapter 4). This ensures that none of the subjects used for training the classifier are included in the crowd image under test. Each crowd image consists of 20 randomly placed subjects, where 9 subjects expressing emotion type A and 11 subjects expressing emotion type B are placed together in 2 separate groups. We define 8 different classes of emotion to represent the ground-truth emotion of the crowd image, namely; the 6 prototypic emotions (joy, surprise, anger, fear, disgust, sadness) as well as neutral and mixed emotion. The emotion of A and B is dependent on the specified ground-truth emotion of the crowd image based on the following algorithm:

```
if ground-truth emotion of crowd ← mixed emotion then
    $A$ ← random emotion
    $B$ ← random emotion
else
    $A$ ← ground-truth emotion
    $B$ ← neutral emotion
```

We generate 3 crowd images for each of the 8 classes of emotion, resulting in 24 crowd images for testing per fold. The total size of our test data across all folds is thus 240 crowd images. An example of a generated crowd image with ground-truth emotion Anger is given below.

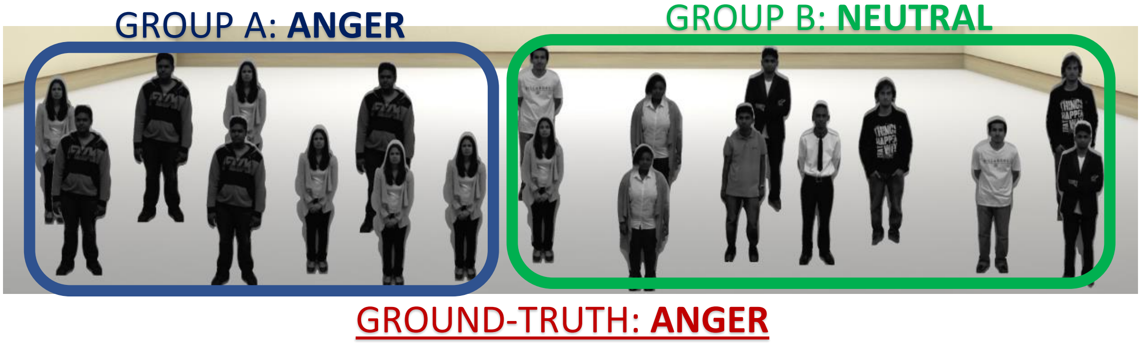


Figure 6.17: Generating a crowd image with ground-truth emotion Anger

The emotion of each of the 24 crowd images is predicted using our proposed algorithm (see Chapter 5) and compared to the known ground-truth emotion. The recognition accuracy for the fold is then found using the equation:

$$\text{Fold Accuracy} = \frac{\text{number of correct predictions}}{\text{24 crowd images}} \times 100\% \tag{6.5}$$

This process is repeated for the remaining 9 folds and the average recognition accuracy is calculated across all 10 folds. The entire 10-fold cross-validation procedure is repeated 10 times and the average overall recognition accuracy is calculated from all 10 runs. The 10-fold cross-validation testing procedure for crowd emotion estimation is summarized in Figure 6.18. In our testing, GLTP was selected for facial feature extraction and combined with an SVM for classification. Optimal parameters were selected from Section 6.3.2. The recognition accuracy obtained when using our proposed crowd emotion estimation algorithm with 8-classes of crowd emotion and a selected crowd threshold of 30% is summarized in Table 6.22.

An average recognition accuracy of 64.6% was achieved from 10 different 10-fold cross-validation tests. Examining the crowd emotion confusion matrix for the tests (Table 6.23), we find that joy, neutral and mixed crowd emotions exhibited a high

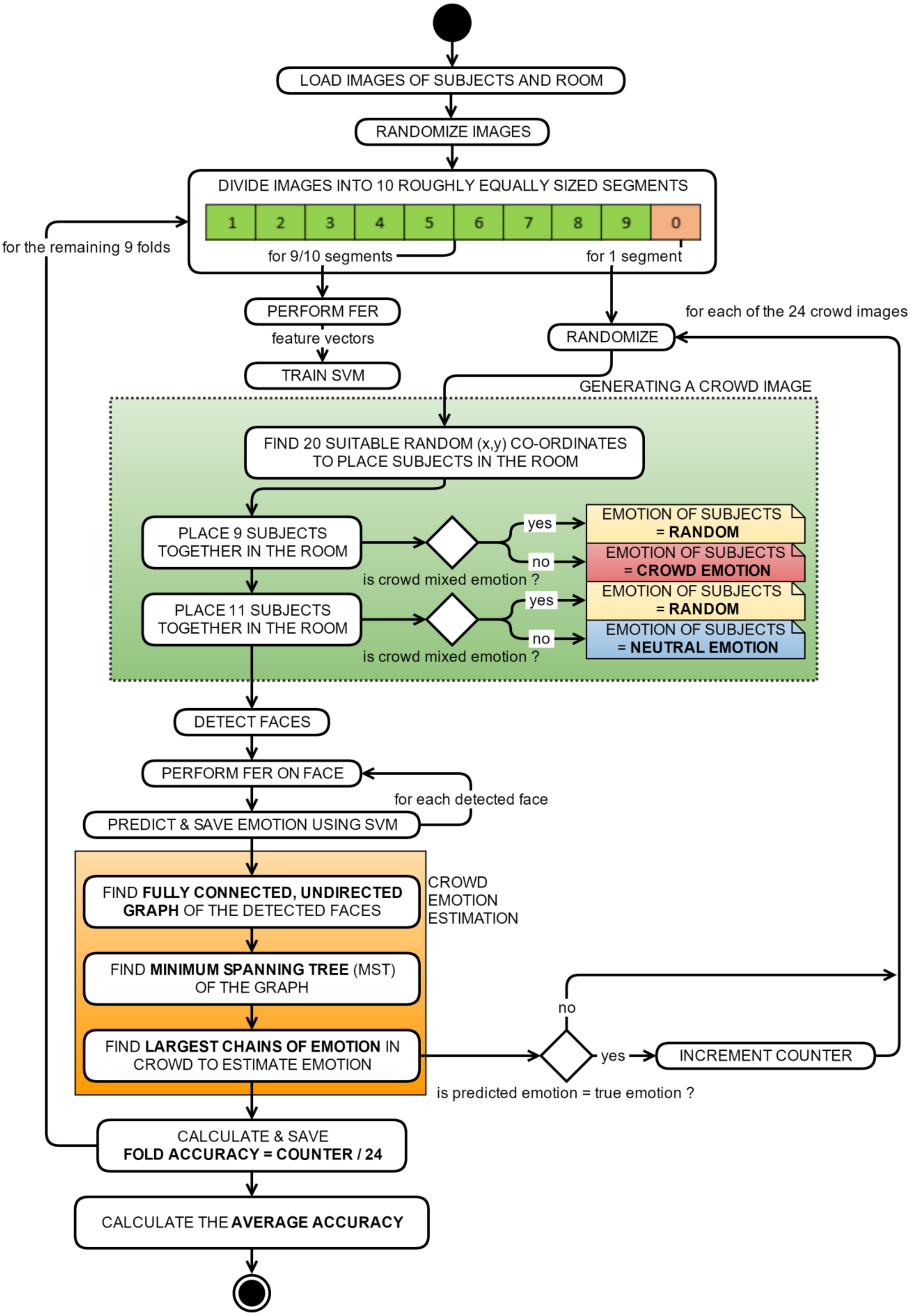

LOAD IMAGES OF SUBJECTS AND ROOM
RANDOMIZE IMAGES
DIVIDE IMAGES INTO 10 ROUGHLY EQUALLY SIZED SEGMENTS
1
2
3
4
5
6
7
8
9
0
for 9/10 segments
for 1 segment
for the remaining 9 folds
PERFORM FER
feature vectors
TRAIN SVM
RANDOMIZE
for each of the 24 crowd images
GENERATING A CROWD IMAGE
FIND 20 SUITABLE RANDOM (x,y) CO-ORDINATES TO PLACE SUBJECTS IN THE ROOM
PLACE 9 SUBJECTS TOGETHER IN THE ROOM
is crowd mixed emotion ?
yes
EMOTION OF SUBJECTS = RANDOM
no
EMOTION OF SUBJECTS = CROWD EMOTION
PLACE 11 SUBJECTS TOGETHER IN THE ROOM
is crowd mixed emotion ?
yes
EMOTION OF SUBJECTS = RANDOM
no
EMOTION OF SUBJECTS = NEUTRAL EMOTION
DETECT FACES
PERFORM FER ON FACE
for each detected face
PREDICT & SAVE EMOTION USING SVM
FIND FULLY CONNECTED, UNDIRECTED GRAPH OF THE DETECTED FACES
FIND MINIMUM SPANNING TREE (MST) OF THE GRAPH
FIND LARGEST CHAINS OF EMOTION IN CROWD TO ESTIMATE EMOTION
CROWD EMOTION ESTIMATION
is predicted emotion = true emotion ?
no
yes
INCREMENT COUNTER
CALCULATE & SAVE FOLD ACCURACY = COUNTER / 24
CALCULATE THE AVERAGE ACCURACY

Figure 6.18: Crowd emotion estimation testing procedure

Table 6.22: Recognition Accuracy (%) for 8-classes of crowd emotion using our proposed crowd emotion estimation algorithm

Cross-Validation Run Number	**Average Recognition Accuracy (%)**
#1	66.3
#2	69.6
#3	63.8
#4	62.9
#5	64.6
#6	63.3
#7	65.0
#8	65.0
#9	61.3
#10	64.2
Overall Average	**64.6 ± 2.1**

Table 6.23: Crowd confusion matrix (%) of cross-validation testing for 8-classes of crowd emotion when using our proposed crowd emotion estimation algorithm

	Joy	**Sur**	**Ang**	**Fear**	**Dis**	**Sad**	**Neu**	**Mixed**
Joy	**97**	0	0	0	0	0	3	0
Sur	0	**62**	0	0	0	0	37	1
Ang	0	0	**5.7**	0	0	0	91.6	2.7
Fear	0	0	0	**58.3**	0	0	38	3.7
Dis	0	0	0	0	**66.3**	0	32	1.7
Sad	0	0	0	0	0	**29**	67	4
Neu	0	0	0	0	0	0	**100**	0
Mixed	0	0	0	0	0	0	1.7	**98.3**

Table 6.24: FER confusion matrix (%) of cross-validation testing for 8-classes of crowd emotion when using our proposed crowd emotion estimation algorithm

	Joy	**Sur**	**Ang**	**Fear**	**Dis**	**Sad**	**Neu**
Joy	**98.8**	0	0.1	0	0	0	1.1
Sur	1.3	**92.3**	0.3	0.2	1.6	0.1	4.2
Ang	0	3.6	**51.3**	0.7	3.1	1.6	39.7
Fear	1.3	3.2	0.3	**85.1**	0	3.9	6.2
Dis	3	0.5	0.1	0.1	**87.9**	0	8.4
Sad	0	1.2	0.1	0.3	0	**71.5**	26.9
Neu	0.2	1	0.3	0.7	0.5	1.1	**96.2**
Average: 85.4%							

degree of recognition accuracy. On the contrary, anger and sadness emotions exhibited a very poor degree of recognition accuracy. These findings share a direct correlation with the chosen method of FER, which achieved an average recognition accuracy of 85.4% on the crowd images. The confusion matrix for FER is given in Table 6.24 and shows that out of the 7 facial emotions on test, anger and sadness achieved the lowest recognition accuracies; being confused to a great extent with neutral emotion. This is supported by our findings in Section 6.3.3, where anger and sadness emotions displayed the most misclassification on test. Due to this misclassification of facial emotion, the size of the group of individuals expressing sadness or anger emotion in the crowd was not able to exceed the required crowd threshold value, thus resulting in incorrect crowd emotion predictions and reduced crowd emotion estimation accuracy.

To determine the true effect of neutral emotion on the accuracy of our proposed crowd emotion estimation algorithm, we repeat the cross-validation tests from above but with neutral emotion excluded. We thus reduce 8-classes of crowd emotion to 7-classes; resulting in a total of 21 crowd images to test per fold. Where before the individuals in group B expressed neutral emotion, they are now placed so that the group exhibits random emotions; none of which exceed the required crowd threshold value. The recognition accuracy obtained when using our proposed crowd emotion estimation algorithm with 7-classes of crowd emotion is summarized in Table 6.25.

With neutral emotion excluded, an average recognition accuracy of 81.3% was achieved from 10 different 10-fold cross-validation tests. This shows a 16.7% improvement compared to when neutral emotion was included. Examining the crowd emotion confusion matrix in Table 6.26, we note that while all emotion classes displayed an improvement in recognition accuracy compared to 8-class testing, in particular, anger and sadness emotions experienced the largest improvement; having increased more than threefold. This is supported by the FER confusion matrix given in Table 6.27, where anger and sadness emotions experienced the most significant

Table 6.25: Recognition Accuracy (%) for 7-classes of crowd emotion using our proposed crowd emotion estimation algorithm

Cross-Validation Run Number	**Average Recognition Accuracy (%)**
#1	80.5
#2	82.9
#3	81.4
#4	81.9
#5	84.3
#6	81.4
#7	80.0
#8	80.0
#9	82.4
#10	78.6
Overall Average	**81.3 ± 1.6**

Table 6.26: Crowd confusion matrix (%) of cross-validation testing for 7-classes of crowd emotion when using our proposed crowd emotion estimation algorithm

	Joy	**Sur**	**Ang**	**Fear**	**Dis**	**Sad**	**Mixed**
Joy	**98.3**	0	0	0	0	0	1.7
Sur	0	**79**	0	0	0.3	0	20.7
Ang	0	0	**51.7**	0	0	0	48.3
Fear	0	0	0	**64.3**	0	0	35.7
Dis	0	0	0	0	**85.3**	0	14.7
Sad	0	0	0	0	0	**91.3**	8.7
Mixed	0.3	0	0	0	0.3	0	**99.4**

Table 6.27: FER confusion matrix (%) of cross-validation testing for 7-classes of crowd emotion when using our proposed crowd emotion estimation algorithm

	Joy	**Sur**	**Ang**	**Fear**	**Dis**	**Sad**
Joy	**99.5**	0	0.4	0	0.1	0
Sur	1.1	**95.5**	0.6	0.6	2.1	0.1
Ang	0	7	**83**	1	4.4	4.6
Fear	1.7	3	0.9	**86.2**	0.1	8.1
Dis	3.2	0.8	1.7	0	**94.2**	0.1
Sad	0.1	2.3	1.3	0.2	0.1	**96**
Average: 93.0%						

increase in recognition accuracy out of the 6 facial emotions on test. With neutral emotion excluded, the average FER recognition accuracy improved by 7.6% from 85.4% to 93%. Further examination of both 7-class and 8-class FER confusion matrices shows that pleasing emotions such as joy and surprise tend to exhibit higher recognition accuracies compared to other displeasing emotions such as anger, fear and disgust, which often get confused between one another. This is evident in Table 6.27, where anger and fear is confused with disgust and sadness.

Now let us consider reducing the 8 and 7 classes of crowd emotion into just 2 classes - positive and negative. Emotions that can be considered pleasing are grouped into the positive class while emotions that can be considered displeasing are grouped into the negative class. For what was previously 7-classes of crowd emotion, we group joy and surprise into the positive class while anger, fear, disgust and sadness are grouped into the negative class. For what was previously 8-classes of crowd emotion, we consider neutral emotion to be non-negative and place it in the positive emotion class. Crowd's of mixed emotion are also considered non-negative and thus classified as positive. We repeat our cross-validation testing on the reduced class set for 2 given scenarios: (1) neutral emotion is included as part of the positive emotion class and (2) neutral emotion is excluded. 24 crowd images are generated per fold, where the first 12 images represent a crowd expressing positive emotion and the final 12 images represent a crowd expressing negative emotion. The individuals in group A are placed so that they are expressing either a random positive emotion (first 12 images) or a random negative emotion (last 12 images), while the individuals in group B are placed so that the group exhibits random positive and negative emotions; none of which exceed the required crowd threshold value. The recognition accuracy obtained when using our proposed crowd emotion estimation algorithm with 2-classes of crowd emotion is summarized in Table 6.28.

An average recognition accuracy of (1) 72.4% (neutral emotion included) and (2) 94.8% (neutral emotion excluded) was achieved from 10 different 10-fold cross-validation tests. These results show an improvement in accuracy of 7.8% compared

Table 6.28: Recognition Accuracy (%) for 2-classes of crowd emotion using our proposed crowd emotion estimation algorithm

Method	**Recognition Accuracy (%)**
With neutral emotion	72.4 ± 2.5
Without neutral emotion	94.8 ± 1.1

Table 6.29: Crowd confusion matrix (%) of cross-validation testing for 2-classes of crowd emotion when using our proposed crowd emotion estimation algorithm

(a) with neutral emotion

	Positive	**Negative**
Positive	**100**	0
Negative	55.2	**44.8**

(b) without neutral emotion

	Positive	**Negative**
Positive	**100**	0
Negative	10.2	**89.8**

to 8-class testing and 13.5% compared to 7-class testing. We note that by excluding neutral emotion from 2-class testing, recognition accuracy improved by 22.4% compared to when it was included. This significant increase in recognition accuracy due to the exclusion of neutral emotion is consistent with our findings during 7-class testing, where we also noted a significant increase in accuracy compared to 8-class testing. The crowd emotion confusion matrix for both 2-class tests is given in Table 6.29. In both cases, all crowd images with positive emotion were correctly predicted; demonstrating that positive emotions may be more easily recognized compared to negative emotions. For the first case, with neutral emotion included, more than half of the negative emotion crowd images on test were misclassified. Some negative emotions, such as anger and sadness, would have been misclassified as neutral emotion (see Table 6.24) causing those crowd images to be incorrectly classified as having positive emotion. For the second case, with neutral emotion excluded, the number of crowd images with negative emotion that were correctly predicted was much higher; resulting in the largest average recognition accuracy achieved on test. Overall, these findings show that greater accuracies can be achieved by combining multiple emotions of a similar type to form a reduced class set, while maintaining the ability to discern negative crowd emotion from positive crowd emotion.

We compare our proposed crowd emotion estimation algorithm to existing crowd

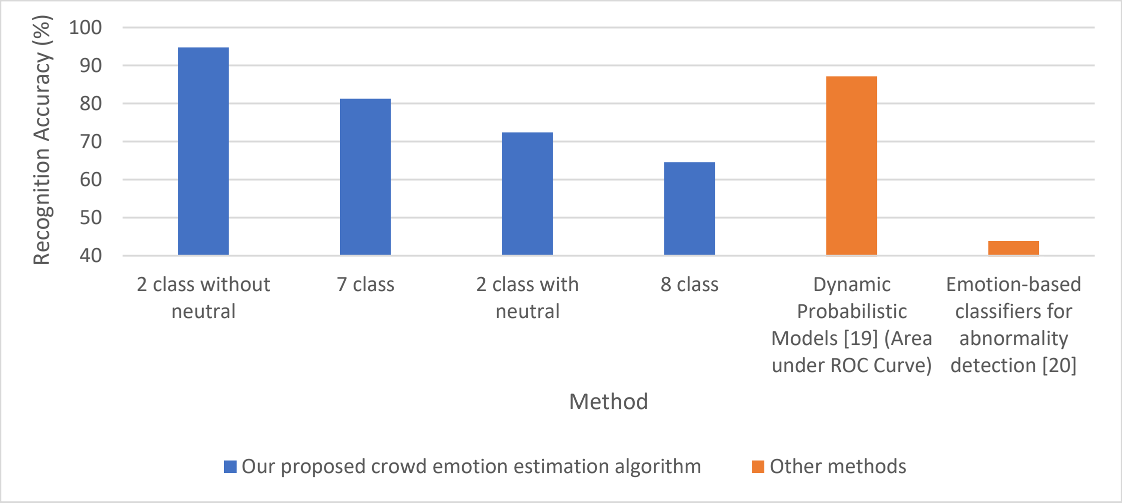


Figure 6.19: Comparison of methods for crowd emotion detection

emotion monitoring techniques identified in Chapter 2. While a direct comparison cannot be made due to differences in the datasets, the testing procedures, and the application of each method, we outline any disadvantages and advantages between methods and where possible compare accuracies. A comparative summary of recognition accuracies for the methods is given in Figure 6.19. In [23], emotion analysis of social media was used to classify crowds into 5 types of behaviour. Although the authors report that their method was successful at classifying a crowd during one mainstream event, no further testing was performed; thus it is difficult to comment on the true accuracy of this method, especially during smaller non-mainstream events. The main disadvantages with this method is (1) it relies heavily on social media with geolocation information and (2) it largely detects emergency situations after they have already occurred. In comparison, our proposed Crowd Monitoring method does not require any additional information apart from the facial expressions of the crowd itself and can detect crowds with a potentially negative emotion before the situation turns to an emergency.

In [20], it was proposed that emotion-based classification of a crowd could be used to better predict crowd behaviour. The authors created a novel crowd behaviour dataset consisting of video sequences for 5 types of crowd behaviour annotated with 6 emotion labels (disgust was excluded) based on the motion of the crowd. Using dense trajectory and SVM classification, emotion descriptions were extracted for each video sequence and mapped to a crowd behaviour. The best accuracy reported for their method was 43.9% using a leave-one-out testing procedure (which typically gives higher accuracies), 20.7% lower than our worst achieved accuracy, although the dataset used in their work was considerably more difficult. Although the authors work represents a novel approach to Crowd Monitoring through the use of crowd emotion, it requires obtaining video sequences of crowds around the apex of their behaviour to be truly effective, which is a complex real-world task. The method is also highly dependent on the type of crowd sequences supplied during the training stage and thus may not work in all environments. In comparison, our proposed method focuses only on 2D static images, which is far more computationally efficient for practical real-world applications. By relying solely on facial expressions for emotion classification, our method should not be greatly effected by changing environments or scenery within the crowd (apart from illumination variation and noise).

In [19], a dynamic probabilistic clustering technique was proposed to model a crowd's response to different events. A simulation model to produce evacuation and panic situations was implemented to test the proposed method. Crowd emotion was classified as either positive or negative based on the clustering together (herding) of individuals within the crowd in response to panic situations. The authors report that a recognition accuracy of 88.6% for correctly detecting positive emotion and 85.8% for correctly detecting negative emotion was achieved using a Receiver Operating Curve (ROC) obtained from 50 simulations. If we were to take the average of these values, we find that the method achieved an average recognition accuracy of 87.2% for both classes of emotion. Ignoring any discrepancies due to differences in testing procedures, we note that the overall accuracy achieved is in the same region ($> 85\%$) as that of our 2-class test results without neutral emotion. However, the authors proposed method is only able to discern positive and negative emotion from panic/evacuation situations; which, depending on how the emotion is defined, may not be a true reflection of negative emotion. While this method is limited to panic and evacuation events, our proposed method can be implemented during multiple types of events for the detection of multiple types of emotion.

6.5.3 Performance

To test the performance of our proposed methodology, we vary the size of crowd while measuring the average time taken to predict the emotion of each crowd image (Core 2 Duo, 2.0GHz, 3GB RAM). The individuals placed in the crowd are selected at random and the results are given in Figure 6.20. The time taken for face detection is excluded from the results as it varies greatly depending on the size of the image. In our testing, each crowd image had a size of 4250 × 1000 pixels. We found that face detection took anywhere from 3.5 seconds (for 1 person) to 8.5 seconds (for 220 people). The results show a linear relationship between crowd size and prediction runtime. We note that for small crowds of 1 to 20 people, prediction takes less than 1 second. On the other hand, for larger crowds of 200 to 220 people, it takes in the region of 12 to 13 seconds for each prediction. We also consider the effect that each step of the prediction has on runtime performance. We identify and analyze the performance of 3 major components of each prediction, namely: (1) Pre-Processing

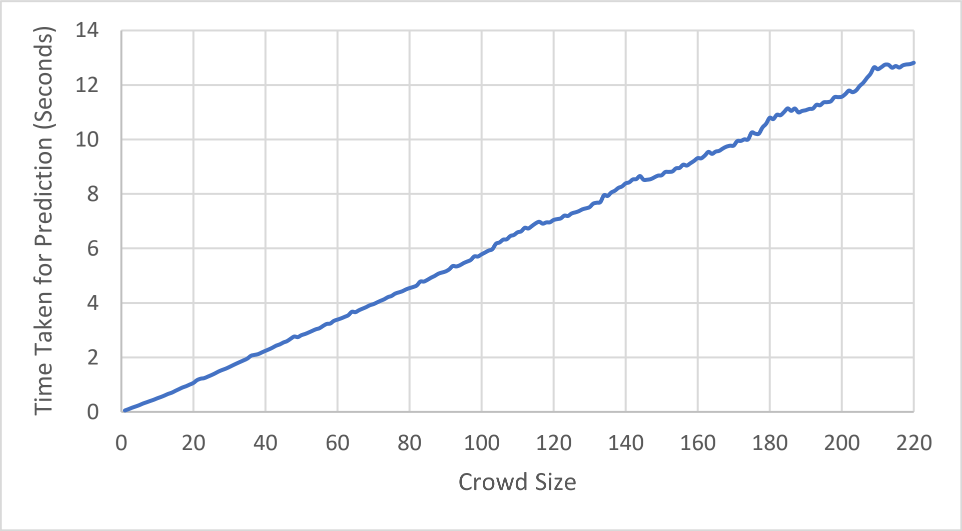


Figure 6.20: The effect of varying crowd size on prediction runtime

of facial images, (2) FER (facial feature extraction using GLTP & classification using SVM), and finally (3) Crowd Emotion Estimation (finding fully-connected undirected graph, finding minimum spanning tree & finding largest chains of emotion in the crowd). The breakdown of results is given in Figure 6.21. From the findings we see that FER constitutes nearly 80% of the time taken for prediction, while the pre-processing of facial images in the crowd constitutes 20.7%. In comparison, our proposed crowd emotion estimation algorithm constitutes just 0.1% of the total time taken for each prediction; demonstrating its impressive efficiency.

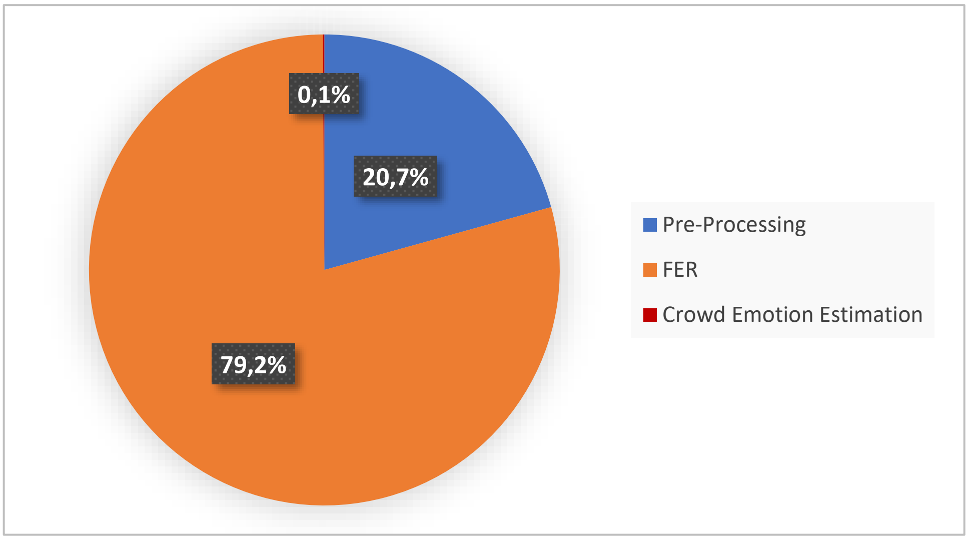


Figure 6.21: Breakdown of prediction runtime

6.5.4 Section Summary

In this section, we confirmed, via extensive testing on generated crowd images with ground-truth emotion, that our proposed crowd emotion estimation algorithm is able to correctly classify a crowd emotion with multiple classes. We found that by excluding neutral emotion and grouping emotions to form a reduced class set, high recognition accuracies were able to be achieved. When testing the performance of our proposed method, it was shown that real-time application is possible and can further be improved by enhancing the efficiency of FER. In a comparison with other methods of Crowd Monitoring, we found that our proposed method offers a viable alternative to existing techniques, provided that facial expressions are unobscured.

Chapter 7

Conclusion

This research has focused on the development of a novel Crowd Monitoring system using Facial Expression Recognition (FER). Existing Crowd Monitoring techniques focus heavily on identifying the movement and density of crowds during panic situations, greatly limiting their ability to predict the intention of a crowd outside of scenes of panic. Our proposed system addresses these limitations by autonomously detecting multiple types of emotion in a crowd in real-time, isolating groups of emotion based on their relative size and weighting in the crowd. This system can be used for many security applications, such as helping to alert authorities to potentially aggressive crowds of people.

In order to validate the effectiveness of the proposed system, a series of cross-validation tests were performed using generated crowd images with known ground-truth emotions. The system was tested for 8, 7 and 2 classes of crowd emotion. The average recognition accuracy for each test was calculated along with it's standard deviation. Confusion matrices were provided for both crowd emotion and FER. The results showed that the system presented was able to accurately predict multiple classes of crowd emotion. The runtime performance of the system was also tested against varying crowd sizes and showed potential for real-time application on moderately sized crowds. The findings show that, compared to existing Crowd

Monitoring techniques, our system is able to accurately and efficiently predict the emotion and thus intent of crowds even in non-panic situations where movement and density information may be incomplete.

A key challenge to producing accurate results is ensuring that the system is able to optimally detect faces within the crowd. There are three major challenges that must be considered when implementing traditional face detection algorithms on crowds, namely: (1) Faces may be highly obscured in large crowds, (2) Not all faces will be frontal facing, and (3) The resolution of the detected face must be adequate for the type of FER being used. The quality and number of faces detected in the crowd will have a direct impact on system accuracy and thus should be optimal. To ensure consistency throughout this research, the assumption was made that all faces in the crowd are unobscured and frontal facing.

7.1 Application

The system presented may be deployed in most indoor/outdoor areas that attract crowds of people; such as airport terminals, shopping centres, sports stadiums, and public rallies. In airport terminals and shopping centres, the system could be used to detect potential criminal threats or danger by identifying groups of individuals expressing fear emotion. At sports stadiums and public rallies, the system could be used to detect disgruntled and disorderly groups of sports fans/protesters by identifying the spread of anger emotion within the crowd. Unlike existing Crowd Monitoring techniques, the system presented here is able to discern between groups of positive emotion and groups of negative emotion in a crowd. At a public rally, for example, this allows the system to distinguish closely packed supportive groups of individuals from closely packed aggressive groups of individuals; which may help authorities to provide a suitable response.

7.2 Future Work

Future research into the system presented is aimed at addressing the current limitations associated with face detection and tracking in densely populated crowds.

- One potential solution to alleviate facial obscurities in large crowds is to use an array of cameras to capture images of the crowd from different angles and positions. The combined number of faces detected from each of these images should show an improvement compared to a single camera-based system.
- In this research, the distance between individuals in the crowd is computed in 2-Dimensional space using x, y co-ordinates to map the location of each individual in the image. By using a 3D camera system as discussed above, the distance between individuals can be computed more accurately in 3-Dimensional space using an x, y, z co-ordinate-based system.
- To account for any variations in facial angle and posture, a non-frontal face detection algorithm could be combined with a non-frontal-based method of FER [95][96] trained with a larger facial image dataset consisting of frontal and non-frontal faces at different angles. In combination with a 3D-based camera setup, this could significantly improve the accuracy of the system for larger crowds.
- To further improve the efficiency of the system, Dimensionality Reduction techniques, such as PCA [44], may be applied during FER to reduce the time taken to predict the emotion of each detected face. The use of PCA with GLTP has previously been discussed in this work.

Bibliography

[1] B. Zhan, D. N. Monekosso, P. Remagnino, S. A. Velastin, and L. Q. Xu, "Crowd analysis: a survey," *Machine Vision and Applications*, vol. 19, no. 5, pp. 345–357, 2008.

[2] D. Benjamin and S. Simon, "The global terror threat in 2016: A forecast," *Combatting Terrorism Center Sentinel*, vol. 9, no. 1, pp. 1–4, 2016.

[3] European Parliament, "Terrorism debate: The threat seems to be increasing in the eu and further afield." `http://www.europarl.europa.eu/news/`, 21 January 2016.

[4] E. Barry, "Protests in moldova explode, with a call to arms on twitter," *The New York Times*, p. A1, 7 April 2009.

[5] P. Ekman, "Strong evidence for universals in facial expressions: A reply to russell's mistaken critique," *Psychological Bulletin*, vol. 115, no. 2, pp. 268–287, 1994.

[6] M. Cikara, A. C. Jenkins, N. Dufour, and R. Saxe, "Reduced self-referential neural response during intergroup competition predicts competitor harm," *NeuroImage*, vol. 96, pp. 36–43, 2014.

[7] J. C. S. J. Junior, S. R. Musse, and C. R. Jung, "Crowd analysis using computer vision techniques," *IEEE Signal Processing Magazine*, vol. 27, no. 5, pp. 66–77, 2010.

[8] T. Li, H. Chang, M. Wang, B. Ni, R. Hong, and S. Yan, "Crowded scene analysis: A survey," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 3, pp. 367–386, 2014.

[9] A. C. Davies, J. H. Yin, and S. A. Velastin, "Crowd monitoring using image processing," *Electronics & Communication Engineering Journal*, vol. 7, no. 1, pp. 37–47, 1995.

[10] B. A. Boghossian and S. A. Velastin, "Motion-based machine vision techniques for the management of large crowds," in *The 6th IEEE International Conference on Electronics, Circuits and Systems*, vol. 2, pp. 961–964, 1999.

[11] E. L. Andrade, S. Blunsden, and R. B. Fisher, "Modelling crowd scenes for event detection," in *18th International Conference on Pattern Recognition*, pp. 175–178, 2006.

[12] S. Ali and M. Shah, "A lagrangian particle dynamics approach for crowd flow segmentation and stability analysis," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–6, 2007.

[13] X. Wu, G. Liang, K. K. Lee, and Y. Xu, "Crowd density estimation using texture analysis and learning," pp. 214–219, 2006.

[14] A. M. Cheriyadat and R. Radke, "Detecting dominant motions in dense crowds," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 4, pp. 568–581, 2008.

[15] V. Rabaud and S. Belongie, "Counting crowded moving objects," in *Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 705–711, 2006.

[16] G. J. Brostow and R. Cipolla, "Unsupervised bayesian detection of independent motion in crowds," in *Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 594–601, 2006.

[17] D. Zhang, C. Tong, Y. Lu, and Z. Liu, "Dominant motions detection in dense crowds based on particle video," *International Journal of Digital Content Technology and its Applications(JDCTA)*, vol. 6, no. 10, pp. 294–301, 2012.

[18] X. Wang, X. Ma, and W. E. L. Grimson, "Unsupervised activity perception in crowded and complicated scenes using hierarchical bayesian models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 3, pp. 539–555, 2009.

[19] M. W. Baig, E. I. Barakova, L. Marcenaro, M. Rauterberg, and C. S. Regazzoni, "Crowd emotion detection using dynamic probabilistic models," in *From Animals to Animats 13*, pp. 328–337, Castelln, Spain: Springer International Publishing, 2014.

[20] H. R. Rabiee, J. Haddadnia, H. Mousavi, M. Nabi, V. Murino, and N. Sebe, "Emotion-based crowd representation for abnormality detection," *CoRR*, vol. abs/1607.07646, pp. 1–7, 2016.

[21] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[22] Y. Wang and G. Mori, "Max-margin hidden conditional random fields for human action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 872–879, 2009.

[23] M. Q. Ngo, P. D. Haghighi, and F. Burstein, "A crowd monitoring framework using emotion analysis of social media for emergency management in mass gatherings," in *Australasian Conference on Information Systems*, (University of South Australia), 2015.

[24] P. D. Haghighi, F. Burstein, H. Li, and C. Wang, "Integrating social media with ontologies for real-time crowd monitoring and decision support in mass gatherings," in *PACIS*, pp. 1–14, 2013.

[25] T. Joachims, "A probabilistic analysis of the rocchio algorithm with tfidf for text categorization," tech. rep., DTIC Document: Carnegie-Mellon Univ Pittsburgh, Dept Of Computer Science, 1996.

[26] A. E. Berlonghi, "Understanding and planning for different spectator crowds," *Safety Science*, vol. 18, no. 4, pp. 239–247, 1995.

[27] B. Loke, "Monitoring group emotions." `www.commit-nl.nl`. Email: b.loke@noldus.nl.

[28] S. Danafar, A. Giusti, and J. Schmidhuber, "Novel kernel-based recognizers of human actions," *EURASIP Journal on Advances in Signal Processing*, pp. 1–11, 2010.

[29] C. Yu and X. Wang, "Human action classification based on combinational features from video," *Journal of Computational Information Systems*, vol. 8, no. 12, pp. 5245–5254, 2012.

[30] S. Mahto and Y. Yadav, "A survey on various facial expression recognition techniques," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 3, no. 11, pp. 13028–13031, 2014.

[31] S. Mishra and A. Dhole, "A survey on facial expression recognition techniques," *International Journal of Science and Research (IJSR)*, vol. 4, no. 4, pp. 1247–1250, 2015.

[32] J. Kumari, R. Rajesh, and K. M. Pooja, "Facial expression recognition: A survey," *Second International Symposium on Computer Vision and the Internet*, vol. 58, pp. 486–491, 2015.

[33] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (Kauai, HI, USA), pp. 511–518, IEEE, 2001.

[34] A. Shaukat, M. Aziz, and U. Akram, “Facial expression recognition using multiple feature sets,” in *2015 5th International Conference on IT Convergence and Security (ICITCS)*, (Kuala Lumpur, Malaysia), pp. 1–5, IEEE, 2015.

[35] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, “Active shape models - their training and application,” *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38–59, 1995.

[36] X. Wang, X. Liu, L. Lu, and Z. Shen, “A new facial expression recognition method based on geometric alignment and lbp features,” in *2014 IEEE 17th International Conference on Computational Science and Engineering*, (Chengdu, China), pp. 1734–1737, IEEE, 2014.

[37] H. Bao and T. Ma, “Feature extraction and facial expression recognition based on bezier curve,” in *2014 IEEE International Conference on Computer and Information Technology*, pp. 884–887, 2014.

[38] T. Ojala, M. Pietikainen, and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[39] P. Suja and S. Tripathi, “Analysis of emotion recognition from facial expressions using spatial and transform domain methods,” *Int. J. Advanced Intelligence Paradigms*, vol. 7, no. 1, pp. 57–73, 2015.

[40] T. Jabid, M. H. Kabir, and O. Chae, “Robust facial expression recognition based on local directional pattern,” *ETRI Journal*, vol. 32, no. 5, pp. 784–794, 2010.

[41] X. Tan and B. Triggs, “Enhanced local texture feature sets for face recognition under difficult lighting conditions,” *IEEE International Workshop on Analysis and Modeling of Faces and Gestures*, vol. 4778, pp. 168–182, 2007.

[42] F. Ahmed and E. Hossain, “Automated facial expression recognition using gradient-based ternary texture patterns,” *Chinese Journal of Engineering*, pp. 1–8, 2013.

[43] S. Bashyal and G. K. Venayagamoorthy, “Recognition of facial expressions using gabor wavelets and learning vector quantization,” *Engineering Applications of Artificial Intelligence*, vol. 21, no. 7, pp. 1056–1064, 2008.

[44] S. S. Meher and P. Maben, “Face recognition and facial expression identification using pca,” in *2014 IEEE International Advance Computing Conference (IACC)*, (Gurgaon, India), pp. 1093–1098, IEEE, 2014.

[45] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences*, vol. 55, pp. 119–139, 1997.

[46] C. W. Hsu and C. J. Lin, “A comparison on methods for multiclass support vector machines,” *IEEE Trans. Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[47] N. S. Altman, “An introduction to kernel and nearest-neighbor nonparametric regression,” *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[48] N. Thomas and M. Mathew, “Facial expression recognition system using neural network and matlab,” in *Proceedings of the IEEE International Conference on Computing, Communication and Applications*, (Dindigul, Tamilnadu, India), pp. 1–5, IEEE, 2012.

[49] Y. Li, Q. Ruan, and X. Li, “Facial expression recognition based on complex wavelet transform,” in *Proceedings of the 3rd IET International Conference on Wireless, Mobile and Multimedia Networks*, pp. 365–368, 2010.

[50] M. J. Lyons, S. Akemastu, M. Kamachi, and J. Gyoba, “Coding facial expressions with gabor wavelets,” in *3rd IEEE International Conference on Automatic Face and Gesture Recognition*, (Nara, Japan), pp. 200–205, IEEE, 1998.

[51] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, “The extended cohn-kanade dataset (ck+): A complete expression dataset for action unit and emotion-specified expression,” in *Proceedings of the Third International Workshop on CVPR for Human Communicative Behavior Analysis*, (San Francisco, CA, USA), pp. 94–101, IEEE, 2010.

[52] T. Kanade, J. F. Cohn, and Y. Tian, “Comprehensive database for facial expression analysis,” in *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, (Grenoble, France), pp. 46–53, IEEE, 2000.

[53] M. Dahmane and J. Meunier, “Prototype-based modeling for facial expression analysis,” *IEEE Transactions on Multimedia*, vol. 16, no. 6, pp. 1574–1584, 2014.

[54] H. Zhou, R. Wang, and C. Wang, “A novel extended local binary pattern operator for texture analysis,” *Inf. Science*, vol. 178, no. 22, pp. 4314–4325, 2008.

[55] F. Ahmed and M. H. Kabir, "Directional ternary pattern (dtp) for facial expression recognition," in *Proceedings of the IEEE International Conference on Consumer Electronics*, (Las Vegas, Nevada, USA), IEEE, 2012.

[56] S. Zhao, Y. Gao, and B. Zhang, "Sobel-lbp," in *2008 15th IEEE International Conference on Image Processing*, (San Diego, CA, USA), pp. 2144–2147, IEEE, 2008.

[57] F. A. Sohel, L. S. Dooley, and G. C. Karmakar, "A dynamic bezier curve model," in *IEEE International Conference on Image Processing 2005*, vol. 2, pp. II – 474 – 477, 2005.

[58] A. Bhadu, R. Tokas, and V. Kumar, "Facial expression recognition using dct, gabor and wavelet," *Int. J. Engg. Innovative Tech. (IJEIT)*, vol. 2, no. 1, pp. 92–95, 2012.

[59] H. Soyel and H. Demirel, "Improved sift matching for pose robust facial expression recognition," in *Proc. IEEE Int. Conf. Automat. Face Gesture Recog.*, pp. 585–590, 2011.

[60] J. J. Bazzo and M. V. Lamar, "Recognizing facial actions using gabor wavelets with neutral face average difference," in *Proc. IEEE Int. Conf. Automat. Face Gesture Recog.*, pp. 505–510, 2004.

[61] E. H. El-Shazly, M. M. Abdelwahab, and R. i Taniguchi, "Efficient facial and facial expression recognition using canonical correlation analysis for transform domain features fusion and classification," in *2015 11th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, (Bangkok, Thailand), pp. 639–644, IEEE, 2015.

[62] J. Yang, D. Zhang, A. F. Frangi, and J. y Yang, "Two-dimensional pca: a new approach to appearance-based face representation and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 131–137, 2004.

[63] H. Hotelling, "Relations between two sets of variates," *Biometrika*, pp. 321–377, 1936.

[64] Z. V. Lambert and R. M. Durand, "Some precautions in using canonical analysis," *Journal of Marketing Research*, vol. 12, pp. 468–475, Nov. 1975.

[65] Y. Lv, Z. Feng, and C. Xu, "Facial expression recognition via deep learning," in *2014 International Conference on Smart Computing (SMARTCOMP)*, (Hong Kong, China), pp. 303–308, IEEE, 2014.

[66] P. Luo, X. Wang, and X. Tang, "Hierarchical face parsing via deep learning," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2480–2487, 2012.

[67] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[68] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layerwise training of deep networks," *NIPS*, 2007.

[69] H. Larochelle and Y. Bengio, "Classification using discriminative restricted boltzmann machines," in *Proceedings of the 25th international conference on Machine learning - ICML 08*, pp. 536–543, 2008.

[70] P. Liu, S. Han, Z. Meng, and Y. Tong, "Facial expression recognition via a boosted deep belief network," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, (Columbus, OH, USA), pp. 1805–1812, IEEE, 2014.

[71] Y. Guo, G. Zhao, and M. Pietikinen, "Dynamic facial expression recognition with atlas construction and sparse representation," *IEEE Transactions on Image Processing*, vol. 25, no. 5, pp. 1977–1992, 2016.

[72] Z. Wang, S. Wang, and Q. Ji, "Capturing complex spatio-temporal relations among facial muscles for facial expression recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 3422–3429, 2013.

[73] M. F. Beg, M. I. Miller, A. Trouv, and L. Younes, "Computing large deformation metric mappings via geodesic flows of diffeomorphisms," *Int. J. Comput. Vis.*, vol. 61, no. 2, pp. 139–157, 2005.

[74] T. F. Cootes, C. J. Twining, V. S. Petrovic, K. O. Babalola, and C. J. Taylor, "Computing accurate correspondences across groups of images," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 32, no. 11, pp. 1994–2005, 2010.

[75] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, 2009.

[76] I. Sobel and G. Feldman, "A 3x3 isotropic gradient operator for image processing, presented at a talk at the stanford artificial project," in *Pattern Classification and Scene Analysis* (R. Duda and P. Hart, eds.), p. 271272, John Wiley & Sons, 1968.

[77] H. Mliki and M. Hammami, "Discriminative regions selection for facial expression recognition," *IJCSI International Journal of Computer Science Issues*, vol. 11, no. 5, pp. 50–57, 2014.

[78] H. Scharr, *Optimal operators in digital image processing.* PhD thesis, Heidelberg University, Heidelberg, Germany, 2000.

[79] Programering, "Opencv2 series of learning notes 8 (image filtering)." `http://www.programering.com`, 2014. Accessed: July 28, 2016.

[80] A. C. Kumar, "Analysis of unsupervised dimensionality reduction techniques," *Comput. Sci. Inf. Syst.*, vol. 6, no. 2, pp. 217–227, 2009.

[81] I. T. Jolliffe, *Principal Component Analysis.* New York, USA: Springer-Verlag New York, 2002.

[82] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (Maui, HI, USA), pp. 586–591, IEEE, 1991.

[83] OpenCV, "Introduction to principal component analysis (pca)." `http://docs.opencv.org/3.1.0/d1/dee/tutorial_introduction_to_pca.html`.

[84] Intelligent Behaviour Understanding Group (iBUG), "300 faces in-the-wild challenge (300-w)." `http://ibug.doc.ic.ac.uk/resources/300-W/`, 2013. Accessed: August 4, 2016.

[85] OpenCV, "Introduction to support vector machines." `http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html`.

[86] B. Solmaz, B. E. Moore, and M. Shah, "Identifying behaviors in crowd scenes using stability analysis for dynamical systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 10, pp. 2064–2070, 2012.

[87] R. Mehran, A. Oyama, and M. Shah, "Abnormal crowd behavior detection using social force model," in *IEEE Computer Vision and Pattern Recognition*, 2009.

[88] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos, "Anomaly detection in crowded scenes," in *IEEE Computer Vision and Pattern Recognition*, 2010.

[89] T. Hassner, Y. Itcher, and O. Kliper-Gross, "Violent flows: Real-time detection of violent crowd behavior," in *3rd IEEE International Workshop on Socially Intelligent Surveillance and Monitoring (SISM) at the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, (Rhode Island), June 2012.

[90] V. Khryashchev, A. Ganin, M. Golubev, and L. Shmaglit, “Audience analysis system on the basis of face detection, tracking and classification techniques,” in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1, (Hong Kong), Mar. 2013.

[91] A. Dhall, “Context based facial expression analysis in the wild,” in *Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on*, Sept. 2013.

[92] R. E. Tarjan, “Minimum spanning trees,” in *Data Structures and Network Algorithms*, vol. 44 of *CBMS-NSF Regional Conference Series in Applied Mathematics*, ch. 6, pp. 72–77, Murray Hill, New Jersey: Bell Laboratories, 1983.

[93] OpenCV, “Open source computer vision.” `http://opencv.org/`.

[94] OpenCV, “Support vector machines.” `http://docs.opencv.org/2.4/modules/ml/doc/support_vector_machines.html`.

[95] Y. Hu, Z. Zeng, L. Yin, X. Wei, J. Tu, and T. S. Huang, “A study of non-frontal-view facial expressions recognition,” in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, (Tampa, FL, USA), Dec. 2008.

[96] W. Zheng, H. Tang, and T. S. Huang, “Emotion recognition from non-frontal facial images,” in *Emotion Recognition: A Pattern Analysis Approach* (A. Konar and A. Chakraborty, eds.), ch. 8, pp. 183–205, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2015.

Appendices

A Consent for publication of images

Consent for the use & publication of images in the Crowd Emotion database

I,

1. __________ *Print Name* __________ *Signature*

2. __________ *Print Name* __________ *Signature*

3. NIKITA PIRTHI *Print Name* Pirthi *Signature*

4. __________ *Print Name* __________ *Signature*

5. KESHAV JEEWANLALL *Print Name* __________ *Signature*

6. Nivaan Krishundutt *Print Name* __________ *Signature*

7. DEVASHNIE BHAGWANDIN *Print Name* __________ *Signature*

8. ADIN ARUMUGAM *Print Name* __________ *Signature*

9. RUVANIA NAICKER *Print Name* __________ *Signature*

10. Kegan Chetty *Print Name* Chetty *Signature*

11. HALALISILE MZOBE *Print Name* __________ *Signature*

12. __________ *Print Name* __________ *Signature*

13. MATTHEW DE NEEF *Print Name* __________ *Signature*

14. BIVEK SINGH *Print Name* __________ *Signature*

15. __________ *Print Name* __________ *Signature*

16. __________ *Print Name* __________ *Signature*

17. __________ *Print Name* __________ *Signature*

18. __________ *Print Name* __________ *Signature*

, hereby give my full consent for the use & publication of the images contained within the Crowd Emotion database.

14/11/2016
Dated