Robust Recognition of Facial Expressions on Noise Degraded Facial Images

by

Munaf Sheikh

A thesis submitted in fulfilment of the requirements
for the degree of
MAGISTER SCIENTIAE
in the Department of Computer Science,
University of the Western Cape

Supervisor: James Connan

Co-Supervisor: Christian Omlin

February 2011

Robust Recognition of Facial Expressions on Noise Degraded Facial Images

Munaf Sheikh

Keywords

Facial expression recognition

Facial action units

Noise

Gabor

Support vector machines

Machine learning

Poisson probability distribution

Gaussian normal distribution

Salt and pepper

Speckle

Abstract

Robust Recognition of Facial Expressions on Noise Degraded Facial Images

Munaf Sheikh M.Sc thesis, Department of Computer Science, University of the Western Cape

We investigate the use of noise degraded facial images in the application of facial expression recognition. In particular, we trained Gabor+SVM classifiers to recognize facial expressions images with various types of noise. We applied Gaussian noise, Poisson noise, varying levels of salt and pepper noise, and speckle noise to noiseless facial images. Classifiers were trained with images without noise and then tested on the images with noise. Next, the classifiers were trained using images with noise, and then on tested both images that had noise, and images that were noiseless. Finally, classifiers were tested on images while increasing the levels of salt and pepper in the test set. Our results reflected distinct degradation of recognition accuracy. We also discovered that certain types of noise, particularly Gaussian and Poisson noise, boost recognition rates to levels greater than would be achieved by normal, noiseless images. We attribute this effect to the Gaussian envelope component of Gabor filters being sympathetic to Gaussian-like noise, which is similar in variance to that of the Gabor filters. Finally, using linear regression, we mapped a mathematical model to this degradation and used it to suggest how recognition rates would degrade further should more noise be added to the images.

February 25, 2011

Declaration

I declare that *Robust Recognition of Facial Expressions on Noise Degraded Facial Images* is my own work, that it has not been submitted for any degree or examination in any other university, and that all the sources I have used or quoted have been indicated and acknowledged by complete references.

Full name: Munaf Sheikh Date: February 25, 2011

Signed:___________________

Acknowledgements

Conducting this research at the University of the Western Cape (UWC) has been a challenging but highly rewarding experience. Over the duration of this research, the following people stand topmost on the list of people who have instrumental in the completion of this endeavour. It is a pleasure to thank those who made this thesis possible:

James Connan – Rhodes University, South Africa – is a lecturer of Computer Science, and my supervisor for this thesis. James has gone out of his way to assist in both a professional and personal capacity. Over the period that I've known him, his support and friendship has been invaluable, and I have grown both professionally and as an individual as a result of his guidance.

Prof Christian Omlin – Middle East Technical University, Northern Cyprus Campus, Turkish Republic of Northern Cyprus – is a Professor of Computer Engineering and my co-supervisor for this thesis. His valuable professional insight into this field has guided the writing of thesis and funneled it's course to the center of a hot research area. It is an honour to have met and worked with him.

Jacob Whitehill – a PhD student at the University of California: San Diego, United States of America – was my mentor in the initial stages of this study. He completed his Masters at the South African Sign Language group at the Computer Science deparment at UWC, and it was the immediate precursor to this research. When he returned to UCSD to do his PhD, Jacob made his support available in a number of ways.

Lastly, I am indebted to Verna Connan (PostGraduate Coordinator, UWC), my wife, and my parents for their limitless assistance and encouragement over the duration of this study.

This research was funded in part by the Telkom/Cisco Center for Excellence for IP and Internet Computing at the University of the Western Cape.

Contents

Glossary

AdaSVM	Adaptive Boosting Selected Feature Representations in Support Vector Machines, 25
ANN	Artificial Neural Network, 16
ASM	Active Shape Model, 41
AU	Action Unit, 3
AUC	Area Under Curve, 65
CKFD	Complete Kernel Fisher Discriminant, 40
CMU	Carneggie-Melon University, 18
FACS	Facial Action Coding System, 3
FER	Facial Expression Recognition, 3
FERET	Facial Recognition Technology, 18
HMM	Hidden Markov Models, 15
ICA	Independent Component Analysis, 15
JAFFE	Japanese Actions Female Facial Effect, 16
LDA	Linear Discriminant Analysis, 15
LFA	Local Feature Analysis, 37

MIT-CBCL	MIT Center for Biological and Computational learning, 18
PCA	Principal Component Analysis, 15
PDM	Point Distribution Models, 15
POFA	Pictures Of Facial Affect, 28
ROC	Receiver Operator Characteristics, 64
SASL	South African Sign Language, 2
SNoW	Sparse Network of Winnows, 33
SNoW-NB	SNoW - Naive Bayes, 33
SVM	Support Vector Machine, 4, 14

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Facial expressions are an integral component of communication. In humans, facial expressions are a means of non-verbal communication of emotions to other humans as well as some animals. Facial expressions serve as adverbs in sentences in everyday communication. Their expressive capability sometimes completely replaces written or even signed languages in the time and efficiency of conveying the intended message.

Automatic recognition of facial expressions is an important component of human computer interaction. Many computer systems now come with voice recognition software.

Facial expressions are crucial for sign language communication. Sign language has two main components: manual hand gestures, and non-manual facial expressions. Sign language is constitutionally recognised as the language of instruction for hearing impaired learners in South Africa. The South African population is growing at a rate of 1.7% [47] per annum making it one of the fastest growing countries in the world. Of this population, 15% are hearing impaired and 17% of this group are fluent in Sign Language - the official communication medium for people with hearing impairments [21]. The reduced ability to communicate with non-hearing impaired people implies a lower quality of living, different learning and working experiences and essentially isolates the hearing impaired community.

The South African Sign Language (SASL) [61] project at University of the Western Cape is an initiative that aims to break down the communication barrier between people with and without hearing impairments. Its plan is to develop technology that translates Sign Language into a Spoken Language. A system that facilitates communication between these two groups of people would thus be invaluable.

This research will focus on aspects of Automated Facial Expression Recognition (FER). Facial expression recognition software could be used to enhance the recognition process. FER systemsvoice could also be used to detect the blackout experienced by fighter jet pilots at high altitudes, and drowsiness of high speed train drivers, carrying hundreds of passengers. Facial expressions are also often used in behavioural science and in clinical practice. Thus, great benefit to many people can be derived from the development of a working FER system.

1.2 Problem Statement

Most of the systems developed for FER cater only for controlled laboratory circumstances in which much of the expression performing is rehearsed and preempted by trained actors. What is not currently known is how well these systems adapt to suboptimal circumstances in which there is image noise and occlusion in the images. Research is needed to determine thresholds for optimal conditions - limits that allow one to say for instance “when 35% of the image is obscured, automated FER still yields results in excess of 80% correctness.”This knowledge is invaluable and can be used to determine the minimum hardware and software requirements for FER.

1.3 Premises

An initial assumption must be made: there exists an expression representation system with which to describe expressions and that it is complete enough to handle the full range of Sign Language expressions. Such a representation system exists in the form of the Facial Action Coding System (FACS). FACS describes a set of individual muscle movements called Action Units (AUs). FACS is described in more detail in Chapter 2.

The second assumption that must be made is that a FER system already exists, and that such a system is capable of detecting action units in facial images with a record comparable to the best classification systems, human or machine. This system must at least be able to detect expressions. Such a system has already been created [79], and in this instance is recreated with optimization for use in this thesis.

Further, it is assumed that there are sufficiently descriptive noise models on which to base experiments. This is necessary to ensure that real-world situations can be recreated in laboratory settings, and that the noise models can be applied to facial images with which to perform experiments.

1.4 Research Questions

To date, the SASL group has focused primarily on developing a holistic FER system. The system in use is currently able to successfully classify expressions. The data set of faces used for training this system has to date consisted only of frontal views. Minimal knowledge exists of the system's generalization powers and how sufficient it will be in meeting the project's needs.

The question that this thesis aims to answer is *how does our facial expression recognition accuracy deteriorate under noisy conditions?* This question can be broken down in to the following subquestions:

1. First, how does the SASL FER system perform with clear images. This needs to be known for control purposes.
2. How does this same FER system perform when trained with clear images and tested on noisy images?
3. Which noises affect the classification accuracy the most and which the least?
4. How does this system perform in reverse: when trained with noisy images and tested with clear images?
5. What is the amount of noise that renders recognition accuracy unacceptable?

1.5 Research Objectives

In order to answer the above questions, the following experiments need to be done.

1. Train a facial expression classifier on clear images and test on clear images to create a point of reference.
2. Train the classifier on clear images and test on noisy images.
3. Train the classifier on noisy images and test on noisy images.
4. Train the classifier on noisy images and test on clear images.
5. Perform the above tests on a set of images with a low amount of noise, and gradually increase the noise level and record the results for each noise level.

1.6 Methodology

The SASL Projects existing FER system will be used with a few optimizations. This FER system uses a Support Vector Machine (SVM) trained with Gabor-filtered images to recognize action units compatible with the FACS. Various noise filters will be applied to the clear images. Several experiments will be performed comparing the recognition accuracy of the FER classifier on various noisy images.

1.7 Thesis Overview

The rest of this thesis is constructed as follows:

FACS will be described in Chapter 2. The decision to use it as the basis representation of facial expressions will be motivated, and FACS will be contextualized amongst other similar systems.

In Chapter 3 a brief review of literature around FER systems will be presented. Emphasis is placed on literature supporting novel feature extraction, novel modelling/classification method and extremely good recognition performance.

Chapter 4 sees an introduction to the SVM - a binary learning and classification system we use to detect facial expressions.

In Chapter 5 the mathematical methods used to process facial images and ready them for use with the SVM will be discussed. The noise models used to distort images is discussed, as well as Gabor filters and how they are used to create feature representations for classification. A brief discussion follows on the metrics employed in evaluating the performance of the classifier.

Chapter 6 sees the experiments conducted and results obtained. All the components of previous chapters are brought together – the Support Vector Machine, FACS, Gabor filters, noise and the results are presented in clear graphs and tables. This chapter sees a summary of all results as well as the average classification results for all expressions as a function of noise. Detailed individual results are viewable in Appendices A and B.

Finally, Chapter 7 concludes this thesis and suggests directions for future research.

Chapter 2

The Facial Action Coding System

The Facial Action Coding System (FACS) [23; 27; 22] is a comprehensive model of muscles on the human face. It was developed by Ekman and Friesen in 1978 and is currently used by several research groups [62; 50] as a backbone in their facial expressions recognition systems for behavioural research. This chapter motivates the exclusive use of FACS for representing facial expressions in the rest of this thesis.

2.1 Introduction

Between the skin and bone on a face is a complex network of muscle, vessels, fat, and various other tissues. Facial muscles can be considered in a mechanical sense as miniature hydraulic pumps attached to a single membrane that, when activated, pull the components together thus causing a warping effect. All faces have the same muscles underlying the tissue and fat, however all faces also have differing amounts of fat and underlying tissue. Facial expressions have been shown to provide information about active state, cognitive activity, temperament, personality, truthfulness and psychopathology [20].

Over eight years a detailed model of the facial muscles was created through the combined use of palpation, study of anatomy, video tape and photographic sessions with subjects[23]. FACS was designed to measure every *visible movement* of the face that occurs as a result of facial muscles contracting. An advantage of this is that it can be used on people who are unaware that they are being monitored. However, FACS cannot measure and is not intended for muscle movements that result in no physical appearance change.

2.2 The Design of FACS

2.2.1 Action Units

The heart of FACS is the *action unit* (AU). An action unit is the minimal unit of facial behaviour. Action units relate to visible movement on a face, which need not necessarily be caused by any one particular muscle. AUs are referenced using a number to simplify coding of faces. In some cases, like AU 13, an action unit corresponds directly to one facial muscle but in others one action unit can reflect a group of visibly indistinguishable muscles. In yet other cases, such as AUs 7 and 8, multiple action units can relate to the same muscle if different parts of the muscle can be contracted independently. In the 2002 edition of FACS [27] there are in excess of 70 action units. Table 2.1 lists the most common action units.

The *FACS manual* contains detailed graphical and textual descriptions of muscular basis for each AU as well as photographic and video examples of facial appearance changes. It also provides instructions on how to perform each AU and criteria to assess the intensity of the AU being performed.

2.2.2 Interdependence between Action Units

Various facial expressions exhibit a combination of one or more action units being contracted and/or relaxed simultaneously. Over 7000 distinct combinations of AUs have been observed in spontaneous behaviour. Combinations can be *additive* or *distinctive*: additive implying that the appearance of individual AUs in a combination is indistinguishable from its appearance when it occurs alone, and distinctive that AUs must combine to create a particular appearance. Combinations can also have *dominant* AUs, where the effect of an AU masks that of the *subordinate* AU. Another that can occur between combinations is *substitutive* combinations where one AU cannot be distinguished from another in the combination. In this case the FACS coder decides which is more relevant.

2.2.3 AU Intensity

The intensity refers to how apparent the action is on a face. An AU can be involved to the following degrees: Uninvolved, Trace, Slight, Marked, Pronounced, Severe, Extreme, and Maximum. Intensity is indicated by the letters A (trace), B (slight), C (marked or pronounced), D (severe or extreme), or E (maximum). The A-E scoring scale is not an equal-interval scale. A, B and E are outlier regions, and most appearance changes fall into the C or D ranges.

Table 2.1: Table 2.1 is a comprehensive list of AUs and their associated actions.

FACS Name	**Muscular Basis**
1	Inner Brow Raiser
2	Outer Brow Raiser
4	Brow Lowerer
5	Upper Lid Raiser
6	Cheek Raiser
7	Lid Tightener
8	Lips Toward Each Other
9	Nose Wrinkler
10	Upper Lip Raiser
11	Nasolabial Furrow Deepener
12	Lip Corner Puller
13	Cheek Puffer
14	Dimpler
15	Lip Corner Depressor
16	Lower Lip Depressor
17	Chin Raiser
18	Lip Puckerer
20	Lip Stretcher
22	Lip Funneler
23	Lip Tightner
24	Lip Pressor
25	Lips Part
26	Jaw Drop
27	Mouth Stretch
28	Lip Suck
38	Nostril Dilator
39	Nostril Compressor
41	Lid Droop
42	Slit
43	Eyes Closed
44	Squint
45	Blink
46	Wink
51-58	Head poisitions
61-66	Eye positions
70	Brows not visible
71	Eyes not visible
72	Lower face not visible

2.2.4 Scoring

FACS must be coded from video. A FACS coder analyses expressions occuring in individual frames and extracts the specific AUs that occurred. With reference to the video, the duration of the AUs presence, its intensity, its onset and offset times are also noted. Actions are *scored* at their *apex* in a particular instance or event. The apex is not the absolute amount of change. It varies from person to person and is the point of greatest deformation in a particular instance. In this research, the apex is the point immediately before the action begins to wane in intensity. FACS scoring is performed by trained experts who make judgements based on their perceptions of video frames. It requires approximately 100 hours to achieve minimal competence in scoring, and typically over two hours to code one minute of video.

2.2.5 Symmetry of Action Units

When an action occurs on only one side of the face it is referred to as *unilateral*. In this case, the letter “R” or “L” is placed in front of the AU number indicating the action occurred on the right or left halves of the face respectively. Unilateral scores that do not indicate which side the action occurred are prepended with “U”, such as “U46” – for a wink. The “A” prefix is used to indicate asymmetry – when the action occurs on both sides of the face but is stronger on one.

2.3 Action Units in Experiments

In this thesis, experiments are performed using AUs for which the *CMU Database*[40] has a minimum of 50 images. This figure comprises sufficient data and has been used in several papers [79]. Table 2.2 shows the list of AUs experimented with.

2.4 FACS, EMFAX, MAX and AFFEX

Because FACS does not encompass emotion expressions, Friesen and Ekman developed EMFACS-7: Emotional Facial Action Coding System [34]. EMFACS has only 33 predefined combinations of *muscular actions*. These combinations encompass all seven prototypical emotions: joy, sadness, anger, surprise, disgust, fear and contempt. Comparatively, FACS is purely descriptive and uses no emotion and other inferential labels. EMFACS is more economical than FACS because it ignores the precise duration of each action and any bilateral symmetry. EMFACS also scores only the emotional activity.

Table 2.2: The AUs in Table 2.2 had sufficient data on which to perform conclusive experiments. This table depicts the AUs and their associated actions.

AU Number	Facial Action
1	Inner Brow Raiser
2	Outer Brow Raiser
4	Brow Lowerer
5	Upper Lid Raiser
6	Cheek Raiser
7	Lid Tightener
9	Nose Wrinkler
11	Nasolabial Furrow Deepener
15	Lip Corner Depressor
17	Chin Raiser
20	Lip Stretch
23	Lip Tightener
24	Lip Presser
25	Lips Part
26	Jaw Drop
27	Mouth Stretch

The Maximally Discriminative Facial Movement Coding System (MAX) [39] is also intended to include only the facial *movements* related to emotion. It does this by making explicit claims that specific combinations of movement are emotional expressions. Certain anatomically distinct movements such as inner and outer-brow raises can not be differentiated in MAX. MAX is therefore less comprehensive than FACS.

While FACS, EMFACS and MAX are objective, AFFEX [38] is a subjective coding system for emotions in children. AFFEX assumes that facial expression and emotion have an exact correspondence. EMFACS, MAX and AFFEX are not as complete as FACS in expression representation capabilities.

Advantages of FACS include:

1. Objectivity: FACS is a description of physical changes in the face, on to which interpretive labels are not applied.

2. Comprehensiveness: FACS codes all expressions that have been observed in over 20 years of behavioural data.

3. Robust link with ground truth. There is over 20 years worth of data describing the relationships between FACS and underlying emotional states.

Shortcomings of FACS include: its inability to describe movement of the pupil, teeth and tongue. It currently takes over 100 hours of training to achieve minimal competency on FACS. A *trained expert* can take up to an hour to score one minute of video [3]. Some upper facial actions require extensive training for humans to score reliably. FACS does not encompass emotion expressions. FACS also lacks temporal and detailed spatial information [28; 25; 42].

2.5 Summary

FACS allows precise specification of morphology and the dynamics of facial movement. While not explicitly included in FACS, Ekman and Friesen proposed that prototypic expressions of emotion could be represented by specific combinations of action units.

FACS has been used to demonstrate fiducial differences between:

1. genuine and simulated pain [56],
2. people telling the truth and lying [26], and,
3. suicidally and non-suicidally depressed patients [36].

FACS is so complete, that FACS-based systems have been incorporated into computer graphic systems to be used for generating realistic expressions for animation characters (Toy Story [41]). FACS continues to be the leading method for measuring facial expressions [24], and is the representation of choice for this thesis.

Chapter 3

Literature review

3.1 Introduction

This chapter presents a brief literature review of the existing FER systems. Emphasis is placed on research that improves the performance of existing classic feature extraction methods. Literature is presented under the following headings and order: Gabor filters, Haar wavelets, Principal Component Analysis, Flow Fields, Holistic Spatial Analysis and Template Matching, Independent Component Analysis, Linear Discriminant Analysis and Point Distribution Models. The chapter concludes with a brief look at complete systems.

3.2 Generic Facial Expression Analysis Framework

Information about the face and expressions thereon are contained in images. An image by itself is a static representation of the face at a particular moment in time. For facial expression recognition, grayscale images are most often used since this greatly reduces dimensionality of pixel information. In some instances, image sequences (videos) are used in place of static images. Video provides temporal characteristics of fiducial features. They show how the face changes as a function of time.

While theoretically complete and independent fields, advances in image processing took hints from breakthroughs in signal processing. The idea that images could be treated as two-dimensional signals allow researchers to apply signal processing ideas to images. Image pre-processing, such as noise removal, pixel intensity normalization, as well as translation, scaling and rotation can be applied to images. Images can be cropped to extract regions of interest

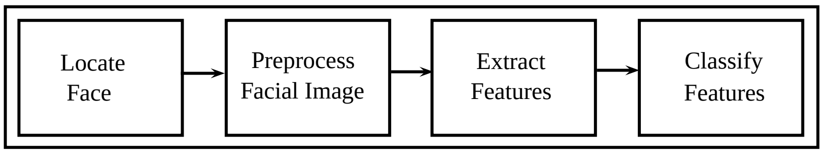


Figure 3.1: Generic Facial Expression Analysis Framework

for more focussed attention. Images are also often normalized prior to classification to reduce the effect of unwanted transformations, particularly if expressions are sensitive to orientation.

FER systems usually take the form of a sequential configuration of processing steps which generally follows the classical pattern recognition framework. Figure 3.1 shows a generalised workflow model of the stages involved. A facial expression, being of the face, requires an image of a face or at least part of a face to be present. This image is then preprocessed. Preprocessing removes noise, improves lighting quality, rotates and generally improves chances of recognising the expression. Features are then extracted that are compatible with the type of machine learning engine used. The machine learning engine is given training data, which is used in several different ways to create a model from which decisions can be made on new, test data. This model is then used to classify test features.

3.2.1 Face Location

In an automated system, face locator algorithms are used to detect and isolate the face. Complex algorithms might be used with CCTV cameras, for instance. These algorithms are able to distinguish between multiple faces in a crowded room with noisy background that might have wall picture frames, windows, and other shiny, round objects. Other, more localized systems, such as webcams, that are focused on close-range images, do not need such extensive algorithms and instead need to only locate the face in images that have less background noise [35].

3.2.2 Preprocess Facial Image

Once the face has been located, image processing takes place. This preprocessing depends on the environment in which the image is taken. For instance, for a camera that is mounted in an open-air location, images taken on a bright, sunny day requires smaller modifications than those taken on a cloudy day. Normalization is a process of cropping, scaling, rotating and colorizing images in order to reduce uncontrollable factors that might throw classification off. Images are often normalized before use [35; 67].

3.2.3 Feature Extraction

Feature extraction methods fall into one of two broad groups: geometry based and appearance based. Geometry based methods are those that use the positions of various key points relative to each other and/or some axis. By measuring the distance moved by each point, one is able to make conclusions about the presence or absence of a particular feature. Appearance based methods classify facial expressions based on facial pixel intensities.

Geometry-based Methods

Geometric models are a top-down approach to FER that employ a model of sorts on which to map facial images. These methods disregard all colour information, except when used to track image points. Models are used in some literature in this form and are sometimes referred to as template-Based Methods. The idea is to create a model, or template, of the face, and warp images onto it. Classification accuracy is determined by the validity of the model. When muscles contract, many factors influence the motion of the skin. Deterministic models make it difficult to accurately account for all of these factors [3] .

Appearance-based Methods

Using a bottom-up approach, the objective is to detect facial features first. These can be isolated using many properties of faces, including edges, intensity, shape, texture and colour. These methods completely ignore the geometric relationships between different key points on the face except where these relationships can be used by image filters. Often referred to as feature-based methods, the aim is to detect invariant features, group them into candidates and verify them. One advantage of feature-based methods is the reduced dimensionality of their input vectors. A disadvantage is that it is difficult to locate facial features if there is a complex background , under differing conditions in lighting, noise or if occlusion is present. Another disadvantage [3] is that it may not be known at the start which image features are the most relevant to classification.

3.2.4 Classification

After the face has been located, processed and its features extracted, the remaining task is to determine the facial expression portrayed in the image. A precursor to this stage is to define a set of categories or classes of facial expression. In literature there are two sets of categories often used. Ekman [62] described 6 *universal expressions*, namely; happiness, sadness, surprise, fear, anger and disgust. There are also muscle-models, of which the Facial Action Coding System (FACS) [27] is one example. The FACS is described in detail in Chapter 2.

3.3 Literature

This thesis builds on both internal and external research done previously. In particular, Whitehill [79] investigated two techniques for increasing both the recognition and computational efficiency of automatic FER. [79] developed a system for detecting FACS action units using Adaboosted Haar features and compared it to a Gabor feature with Support Vector Machine (SVM) approach. This thesis extends that research by:

- testing the Whitehill and Omlin system[79] on a larger set of action units,
- introducing four types of noise and testing for deterioration in recognition accuracy.

At the time of writing of this thesis, there are in excess of 1000 literature works on FER. There are several tutorials and introductory works that describe in detail the work that exists as well as future scope for the various avenues of research in FER. This chapter presents selected works that are useful in explaining various newer facets and adaptations of older, more common methods.

This section is broken down into the following 10 subsections. Each of the first 9 subsections represents a *feature extraction method* that can be used to train a classifier. In accordance with the theme of recognising facial expressions robustly, emphasis is placed on research that improves the performance of the existing classic feature extraction methods. The last subsection presents a brief look at complete systems.

Methods for improving Gabor filters, being the feature extraction method of choice in this thesis, are presented first and in most detail (see Section 3.3.1). Further technical details of Gabor filters can be found in the Mathematical Methods chapter – Chapter 5.2. A brief look at the Haar wavelets (see Section 3.3.2) follows, which were also analyzed by Whitehill [79].

Thereafter, the most novel extensions to popular FER feature extraction methods are reviewed. In the order that they are presented, they are: Principal Component Analysis (PCA) (see Section 3.3.4), Flow Fields (see Section 3.3.5), Holistic Spatial Analysis and Template Matching (see Section 3.3.6), Independent Component Analysis (ICA) (see Section 3.3.7), Linear Discriminant Analysis (LDA) (see Section 3.3.8) and Point Distribution Models (PDM) (see Section 3.3.9). The chapter is concluded with a brief look at complete systems (see Section 3.3.10).

As will be seen, it is useful to recognize that these sections are highly overlapping. Researchers have taken various feature extraction methods and combined them with others, or used them in different permutations with machine learning algorithms while attempting to improve previous results. As an example, PCA is often used both in feature extraction – with for instance, Hidden Markov Models (HMM) as the classifier – and in classification – where Gabor filters are, for instance, the feature extraction method.

3.3.1 Gabor Filters

Technical details of Gabor filters can be found in Chapter 5.2. This subsection addresses some important contributions made to FER using Gabor Filters:

- *channels* of Gabor filters,
- methods for *fusing* multiple Gabor filters,
- selecting Gabor features for SVMs using *Adaboost*,
- local, global and semi-global Gabor features, and,
- the results of an experiment that suggests using gray images as well as Gabor filtered images improves classification performance.

Gabor Channels

Liu and Wang [57] from the University of Science and Technology of China analyzed the contributions that *channels* – a group of Gabor filters with the same scale and orientation – have on facial expression recognition.

They are able to analyze the contribution that various scales and orientations have on recognition rate by grouping the Gabor filters into their respective channels and analyzing the contribution made. In total there are 13 channels that corresponded to 5 scales and 8 orientations. By combining the features from channels that produced good recognition rates, an improvement was seen in the overall recognition rate of the system.

To recognise facial expressions, they needed a method to *fuse* multiple Gabor features. During training, a reliability table was produced, which describes the contribution of different channel features to different facial expression. Then, for each test image, PCA was used to recognize the facial expression on each of the 13 Gabor channel-feature vectors. A three-layer artificial neural network (ANN) with 13 inputs, a hidden layer of 10 TAN-SIG neurons, and an output layer with 7 nodes was used to produce the result.

ANNs consist of three layers. The first layer (input layer) consists of n input nodes where n is the dimension of the feature vector. The second layer (hidden layer) consists of $\frac{(t+c)}{2}$ neurons where c is the number of the classes. The activation function used for this layer is the sigmoid function. The third layer (output layer) consists of c neurons, which are activated by a linear function.

They define $R(c,e)$ as the *reliability* of correctly recognizing facial expression e using the Gabor features of channel c, and $r(c,e)$ as the estimation of correctly recognizing facial expression e using Gabor features of channel c. Figure 3.2 shows the contributions of one such configuration of Gabor channels.

They listed results of NN fusion, Gabor PCA as well as of two other fusion techniques – *Max fusion* and *Sum fusion*

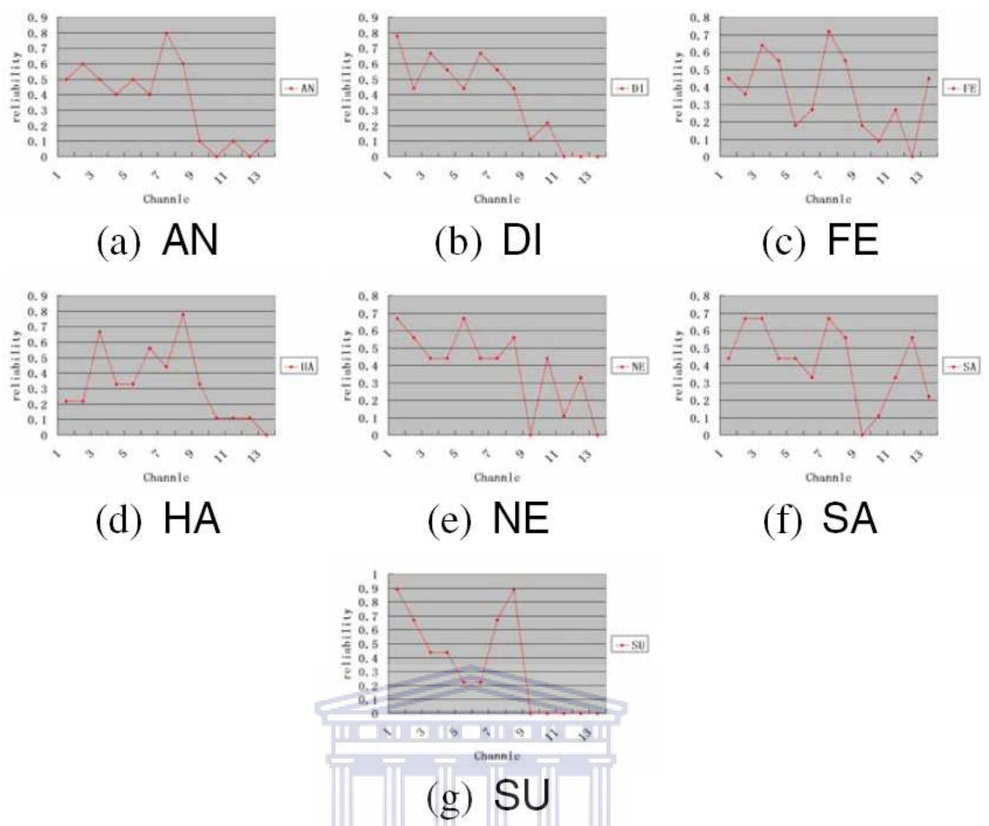


Figure 3.2: Graphs mapping reliability against number of channels used for each prototypical expression: Anger, Disgust, Fear, Happiness, Neutral, Sadness, Surprise. Liu and Wang [57].

– about which they go into more detail in their publication. On the JAFFE [58] [1] database, the results showed that fusion techniques consistently outperformed the Gabor-PCA combination, and that NN Fusion always outperformed Max Fusion and Sum Fusion. With NN Fusion, they are able to push recognition rates up from 60% (Anger), 80% (Fear), 56% (Happiness) and 56% (Sadness) using Gabor PCA, to 95%, 100%, 100% and 90% respectively, all of which are substantial improvements.

Finding the Optimal Specifications of Gabor Based Facial Landmark Classifier

Fasel and Bartlett [30] perform subject-independent experiments with the aim of finding optimal specifications for a Gabor filter based facial-landmark classifier. Using the A' statistic – see Section 5.3 – they conclude that current Gabor filter bank systems are overly complex. They show that single frequency bands return better performances than multiple frequency bands. They conclude that the best performance may be achieved by concentrating a large number

[1]The JAFFE database contains images of subjects posing for seven prototypical expressions, namely, anger (AN), disgust (DI), fear (FE), happiness (HA), neutral (NE), sadness (SA) and surprise (SU). The JAFFE database has ten subjects posing for at least six of the expressions, and two images of each expression for all subjects are used as training examples.

Table 3.1: Best A' for several example combinations of number of orientations and frequency for pupil detection. Fasel and Bartlett [30].

pixels/cycle	6	10	18	22	28	38	44
A'	0. 7038	0. 7507	0. 8604	0. 8538	0. 8603	0. 8767	0. 8955
orient	2	2	2	8	8	8	8

Table 3.2: Best A' for several example combinations of number of orientations and frequency for philtrum detection. Fasel and Bartlett [30].

pixels/cycle	6	14	22	28	34	38	44
A'	0. 5332	0. 7794	0. 8615	0. 8650	0. 8781	0. 8734	0. 8664
orient	2	2	2	2	8	8	8

of orientations on very low frequency carriers. The number of orientation bands can be as many as 8, while the spatial frequencies should be in the order of 5 to 8 iris widths.

They train a nearest neighbor classifier on the FERET [63] face database. Their process for the experiment is to apply a bank of Gabor filters at desired fiducial points, and lastly, compute the Euclidean distance from the output at that fiducial point to the output in the training database that it is closest to. In total, 576 permutations of frequency vs number of orientations, each cross-validated 5 times with different different images, produces 2880 experiments.

Best results for pupils were obtained using 8 orientations and carriers of 32 pixels/cycle (about 4 iris widths). The best detector was with 55 pixels/cycle (about 5.5 iris widths). They found that as the peak frequency decreases, the optimal number of orientations increases. See Table 3.1 and Table 3.2.

Adaboosting Gabor Features

Littlewort et al. [53] designed a fully-fledged system that automatically detects faces in a video stream and codes them with respect to the 7 prototypical expressions. Tested on the Cohn-Kanade dataset[2][40], their best performance was 91.5% correct – the number of correctly classified images from the total, as a percentage. They currently have a version of the system running online [55] that has to date yielded 80% classification accuracy on unconstrained images from the web.

The face finder they used was first developed by Fasel and Movellan and locates the face in each frame automatically. This face finder has a success rate of over 90% when tested on various unprocessed online face-labelled

[2] The data in the Cohn-Kanade database are of 100 university students ranging in age from 18 to 30 years. There are 65 female, and 35 male subjects. There are 15 African-American and 3 Asian or Latino subjects. The subjects pose in a controlled lab environment directly facing the camera and make each expression starting from the neutral pose.

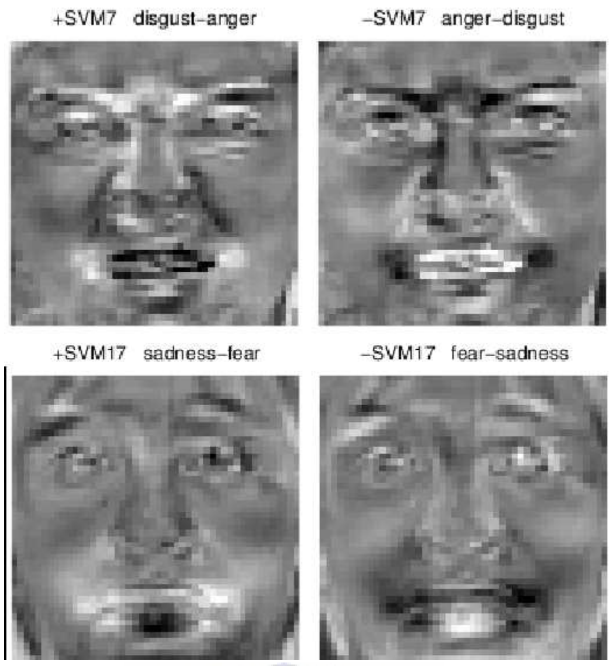


Figure 3.3: The right column is the image negative of the left. Littlewort et al. [53].

databases such as the MIT-CBCL [3] [29]. The product of the face finder is a 48 by 48 pixels *patch* in which the face is centered so that the inter-pupiliary distance is approximately 24 pixels. This system uses 313 sequences from 90 subjects. The images are converted to a Gabor magnitude representation with a bank of 40 Gabor filters at 8 orientations and 5 spatial frequencies, and normalized to unit length.

Because of the small size of training datasets, it might be the case that classifiers learn to predict based on physical differences in the subjects. Often subjects are not fully able to produce a particular muscle contraction properly and independent of others. Thus datasets are limited in the variety of expressions that subjects produce. In this database, certain subjects display certain sets of emotions so personal differences could be predictive of the different expression categories. There are a few ways around this:

- a strategy involving splitting the classification into two stages, the first using identity-matched pairs of expressions and the second stage training on all available data. SVMs are used for the pairwise classifiers in stage 1. The output of stage one has less identity information than its input and this is used as input for stage 2, in which they convert the input into a probability distribution over 7 expression categories. Their paper lists a few approaches, some which failed and others that are successful. Their best results came from combining three sets of pairwise classifiers. Training Gaussian SVMs on Gabor filtered upper face images, lower face images and on

[3] The MIT [2] face recognition database contains high resolution pictures, including frontal, half-profile and profile views of face images of 10 subjects. There are 200 images per subject, with varied illumniation, pose (up to 30^o rotation in depth) and the background.

whole face images, followed by multinomial logistic regression in stage 2 yielded 91.5% recognition accuracy when generalizing to new subjects.

- train classifiers to discriminate one emotion from everything else. Of the two methods, this method provides the largest data set per classifier and dilutes identity effects the most. Their best results of 89.5% for this experiment came from using a linear kernel SVM on full Gabor representations of whole face images.

They also compared SVM performance with that of Adaboost. Adaboost, in its context here, provides guidance in feature selection by selecting the features most informative to test at each step in a cascade. By using only the Gabor features that are selected by Adaboost for training SVMs, processing speed was increased by up to 30 times. Adaboost reduced the input space dimension from its original size of 92160 to 538, and while prediction performance remained essentially unchanged, the processing speed displayed an impressive boost (see Figure 3.3).

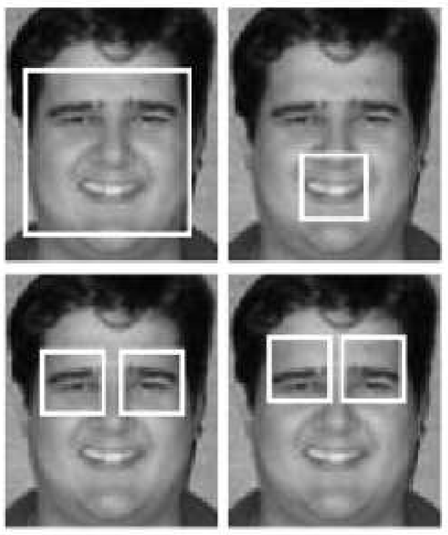
Local, Global and Semi-Global Gabor Features

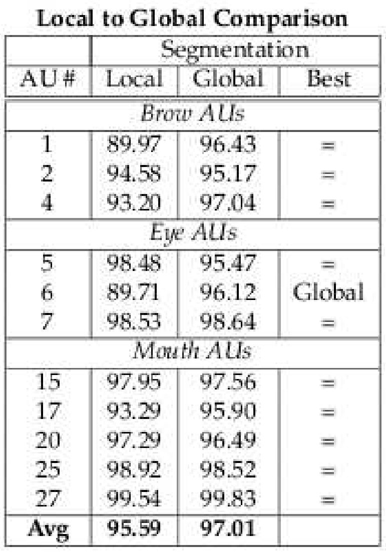
Whitehill and Omlin [77] compare three strategies for AU recognition. They compare local analysis in which classification is performed over a small area around the center of the AU, semi-global analysis in which the half-face in which the AU being classified is located, and global analysis in which the entire face is used in classifying an AU. Whitehill and Omlin use SVMs and mean pixel intensity values in all their experiments.

Comparing the three strategies serves to inform us whether local processing and classification of AUs results in higher AU recognition accuracy than global classification. They used the CMU Database, and AUs are used if they have at least 40 images in the database.

In order to eliminate the chance that classifiers would learn physical characteristics of subjects, they split their image database in half randomly, by subject. This ensures that no subject appears in both a training set and a test set. For each experiment they trained on both linear and radial basis function (RBF) kernels. They used three strategies, and eleven classified AUs.

Their results showed that for AU 1, AU 4, AU 6 and AU 7, global analysis was superior, but for the remaining action units – AU 2, AU 5, AU 15, AU 17, AU 20, AU 25 and AU 27 – the analysis method was dependent on the inter-correlation factor between two AUs. Figure 3.4 shows global (top-left) segmentation and local segmentations of the mouth, eyes and brow regions (remaining images). The table in Figure 3.4 lists some comparative results. They also showed that in cases where high inter-correlation occurred, global analysis proved superior, especially where the dependant AUs are in the opposite half of the face.





Local to Global Comparison

	Segmentation		
AU #	Local	Global	Best
Brow AUs			
1	89.97	96.43	=
2	94.58	95.17	=
4	93.20	97.04	=
Eye AUs			
5	98.48	95.47	=
6	89.71	96.12	Global
7	98.53	98.64	=
Mouth AUs			
15	97.95	97.56	=
17	93.29	95.90	=
20	97.29	96.49	=
25	98.92	98.52	=
27	99.54	99.83	=
Avg	**95.59**	**97.01**	

Figure 3.4: Clockwise from top-left: Global segmentation and local segmentations of the mouth, brow and eye regions. Whitehill and Omlin [77].

Duchenne Smiles

Littlewort et al. [52] test the Gabor-SVM combination on the task of differentiating between Duchenne vs non-Duchenne smiles. Duchenne smiles are genuine smiles and are characterized by contraction of the orbicularis oculi - the sphincter muscles that circle the eyes. Their system yields excellent results of 90% - far higher than previous systems of the time.

The system was tested on three image sets: J. Cohn's Pittsburgh set, P. Ekman and J. Hager's UCSF set, and C. Riley's BBC set. A standard Gabor bank was used that consists of 5 spatial frequencies and 8 orientations. Images are

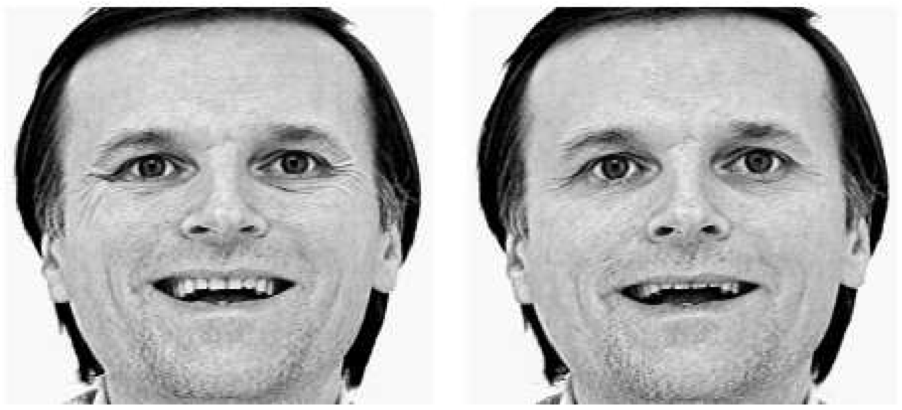


Figure 3.5: Left: Duchenne smile, Right, Non-Duchenne smile. Littlewort et al. [52]

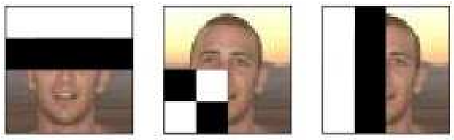


Figure 3.6: Haar Wavelets. Whitehill and Omlin [78]

tested for the presence or absence of FACS AUs 6 and 12. Having both 6 and 12 present indicates that the image is Duchenne, while having only 12 present indicates the image is non-Duchenne.

The results for a RBF SV kernel on gray-scale distance was 87%, and 90% for a linear combination of SVM outputs of plain gray and Gabor filtered SVMs. The best individual SVM performance was 86%. Their extensive experiments showed that linear SVM kernels do not perform as well as polynomial or RBF kernels, that Gabor filtered images outperform unfiltered images, and that Gabor representations allow the use of simpler SVM Kernel functions. Another conclusion was that linear combinations of several experts can outperform individual SVMs when both gray-scaled images and Gabor filtered images are used.

3.3.2 Haar Wavelets

There is growing interest in the use of Haar wavelets for fiducial feature extraction. Few CPU instructions are needed in computing Haar filters when using an *integral image* [73]. The two-dimensional Haar decomposition of a square image of length n pixels is exactly complete[4]. The first wavelet is the mean pixel intensity value of the whole image. The remaining wavelets are calculated as the difference in mean intensity values of horizontally, vertically or diagonally adjacent squares. This section shows that:

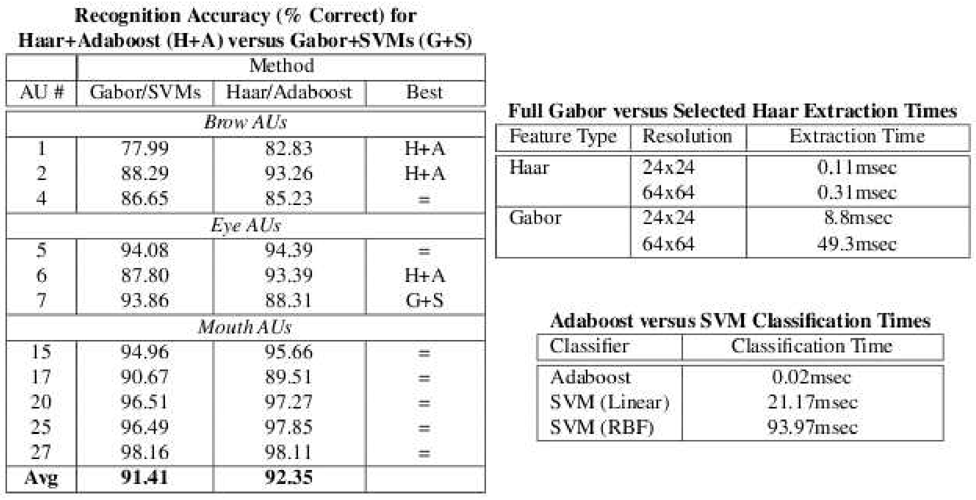
- *Haar+Adaboost* is more than twice as fast as Gabor+SVM but comparable in recognition accuracy,
- Gabor+SVM generally yielded marginally better results than *Haar+SVM*,
- *AdaSVM* is faster and yields better results than SVMs, Adaboost and Gentleboost.

Gabor Wavelets vs Haar Wavelets - Whitehill et al.

Whitehill and Omlin [78] combine Haar features and the Adaboost algorithm in creating a FACS Action Unit recognition system. They compared both the speed and accuracy of their system with the Gabor-SVM method. Their results showed that the Haar-Adaboost technique matches the Gabor+SVM in accuracy but is more than twice as fast as the Gabor-SVM method in classification speed.

The Adaboost boosting algorithm is used as a means of feature selection by constructing a weak classifier out

[4]It has n^2 distinct Haar wavelets.



Recognition Accuracy (% Correct) for Haar+Adaboost (H+A) versus Gabor+SVMs (G+S)

	Method		
AU #	Gabor/SVMs	Haar/Adaboost	Best
Brow AUs			
1	77.99	82.83	H+A
2	88.29	93.26	H+A
4	86.65	85.23	=
Eye AUs			
5	94.08	94.39	=
6	87.80	93.39	H+A
7	93.86	88.31	G+S
Mouth AUs			
15	94.96	95.66	=
17	90.67	89.51	=
20	96.51	97.27	=
25	96.49	97.85	=
27	98.16	98.11	=
Avg	**91.41**	**92.35**	

Full Gabor versus Selected Haar Extraction Times

Feature Type	Resolution	Extraction Time
Haar	24x24	0.11msec
	64x64	0.31msec
Gabor	24x24	8.8msec
	64x64	49.3msec

Adaboost versus SVM Classification Times

Classifier	Classification Time
Adaboost	0.02msec
SVM (Linear)	21.17msec
SVM (RBF)	93.97msec

Figure 3.7: Comparing Haar+Adaboost with Gabor+SVMs using number of images correctly classified expressed as a percentage). Whitehill and Omlin [78].

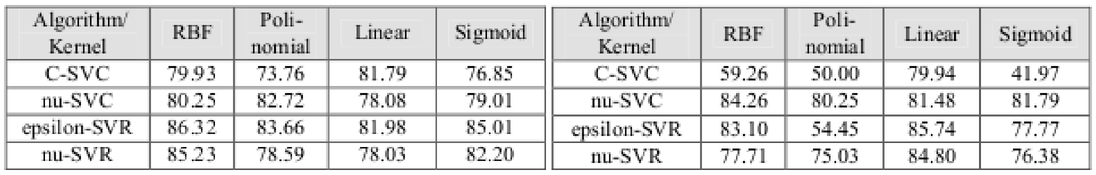
of each Haar feature. A threshold-based binary classifier is created from each Haar feature to minimize the training errors. During each round of boosting, the single best weak classifier for that round is chosen. The final result of boosting is a strong classifier whose output is computed as a thresholded linear combination of the weak classifiers.

From normalized images of faces, Whitehill and Omlin compared system performance when used with 24X24-pixel wide squares and 64X64-pixel wide regions. Three squares were cropped from the eyes and mouth regions to be the input of the Adaboost-based feature selection process. An integral image was used in extracting features from images. For each AU, 500 Haar features are selected using Adaboost and each one fed to its corresponding weak classifier. How the weighted sum of the weak classifiers compares to the strong classifier's threshold determines the strong classifier's classification label for that AU.

The Gabor+SVM method used a filter bank of 5 frequencies and 8 orientations, and Gabor responses were extracted at all points in all filtered images. Three brow AUs - AU 1, AU 2, AU 4 -, three eye AUs - AU 5, AU 6, AU 7 - and 5 mouth AUs - AU 15, AU 17, AU 20, AU 25, AU 27 - are used in their experiments. Haar feature extraction was 80 times faster than Gabor feature extraction for images sized 24X24, and nearly 160 times faster for images sized 64X64. The Adaboost strong classifier is at least 3 orders of magnitude faster than the linear SVM kernel.

Gabor Wavelets vs Haar Wavelets - Batista et al.

Photogenic people, as defined by the Oxford Dictionary, are people who are capable of being photographed attractively. Photogenic photos are most often those in which the subject has either a neutral facial expression or the subject is smiling. A non-photogenic photo is one in which the subject appears sad, surprised, angry, disgusted or frightened.



Algorithm/ Kernel	RBF	Poli-nomial	Linear	Sigmoid
C-SVC	79.93	73.76	81.79	76.85
nu-SVC	80.25	82.72	78.08	79.01
epsilon-SVR	86.32	83.66	81.98	85.01
nu-SVR	85.23	78.59	78.03	82.20

Algorithm/ Kernel	RBF	Poli-nomial	Linear	Sigmoid
C-SVC	59.26	50.00	79.94	41.97
nu-SVC	84.26	80.25	81.48	81.79
epsilon-SVR	83.10	54.45	85.74	77.77
nu-SVR	77.71	75.03	84.80	76.38

Figure 3.8: The table on the left shows the recognition rates using Gabor filters while the table on the right shows recognition rates using Haar wavelets. Batista and Gomes [5]

The classification problem, as per the SVM, thus becomes one of differentiating neutral and happy expressions from sadness, surprise, anger, disgust and fright.

Batista and Gomes [5] train an SVM to separate labeled photogenic from non-photogenic Gabor-filtered face images. They use various combinations of algorithms – including C-SVC, ν-SVC, ϵ-SVR, ν-SVR – and kernels – including RBF, Polinomial, Linear and Sigmoid – and compare the results for all. Their work shows that out of 324 images, their highest recognition rate is the 86.32% produced by the ϵ-SVR algorithm using the RBF kernel.

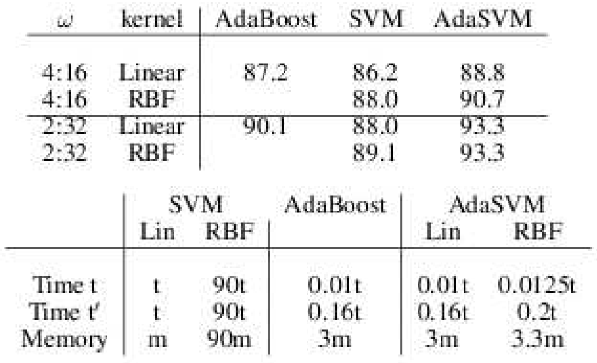
Batista and Gomes [5] performed three experiments. The first using Gabor filters and the second using Haar wavelets. The first two experiments performed 10-fold cross-validation but in the third experiment, the image database was split 75% for training and 25% for testing, and people contained in the training set are not included in the test set. Each experiment cross-tested four algorithms with four kernels. The results for the first and second experiments are shown in Figure 3.8. The third experiment yielded a best result of 75% of images classified correctly with C-SVC algorithm and a Polynomial SVM kernel.

From the results, it is possible to conclude that the best results are obtained using the epsilon-SVR algorithm and a RBF kernel, and that SVMs trained with images pre-processed with Gabor filters generally yielded better results than those trained with Haar wavelet pre-processed images. Using a confusion matrix they show that the images that are mistaken as photogenic displayed a facial expression that portrayed some degree of anger or fear. Non-photogenic images of surprise and sadness were classified as photogenic as surprise and sadness. The images that displayed some degree of sadness or fear were most often classified incorrectly as non-photogenic.

Adaboost, Gentleboost and AdaSVMs

Littlewort et al. [54] present a system that finds faces and codes the 7 prototypical emotions of facial expressions in real time. A cascade of feature detectors trained with boosting techniques forms the face finder. The expression recognizer uses a combination of Adaboost and SVMs. The generalization performance exceeds 90%.

All detected faces are then rescaled and converted into Gabor magnitude representations using a bank of Gabor filters at 5 spatial frequencies and 8 orientations. Training of 7 SVMs then took place - one for each prototypical



ω	kernel	AdaBoost	SVM	AdaSVM
4:16	Linear	87.2	86.2	88.8
4:16	RBF		88.0	90.7
2:32	Linear	90.1	88.0	93.3
2:32	RBF		89.1	93.3

	SVM		AdaBoost	AdaSVM	
	Lin	RBF		Lin	RBF
Time t	t	90t	0.01t	0.01t	0.0125t
Time t′	t	90t	0.16t	0.16t	0.2t
Memory	m	90m	3m	3m	3.3m

Figure 3.9: A comparison between AdaBoost, SVM and AdaSVM methods. Littlewort et al. [54]

emotion. Each SVM was trained to discriminate each emotion from everything else. The emotion displayed for the test image is decided by the SVM with the highest output. The system tested linear, polynomial and RBF kernels with Laplacian and Gaussian basis functions. The best results came from linear and RBF kernels using a unit-width Gaussian.

Adaboost is then applied. Classifiers are trained until a gap proportional to the widths of the two distributions separates the distributions for positive and negative samples. 538 filters are thus selected for 48X48 images with 5 spatial frequencies.

Littlewort et al. [54] detail an approach where Gabor features chosen by Adaboost are used as a reduced representation for training SVMs. They called this method *AdaSVM* - Adaptive Boosting Selected Feature Representations in Support Vector Machines. This method outperformed SVMs by 2.7% . The approach indicates that increasing the number of spatial frequencies from 5 to 9 could potentially improve performance. The tables in Figure 3.9 shows both a accuracy and time and memory comparison between the methods. At 93% and 97% accuracy on two different public databases, AdaSVM outperformed Adaboost and SVMs. AdaSVM also displayed a considerable speed advantage.

3.3.3 Feature Points

Feature points on a facial model are locations that are geometrically mapped to a fixed point on that model. The points are often tracked as the model warps as facial expressions are performed. Feature points can be tracked either manually or automatically. This section shows that:

- Gabor wavelet coefficients at feature points significantly improves results compared to using geometric data alone,
- Feature point tracking performs well for facial regions with less movement than occurs around the mouth regions.

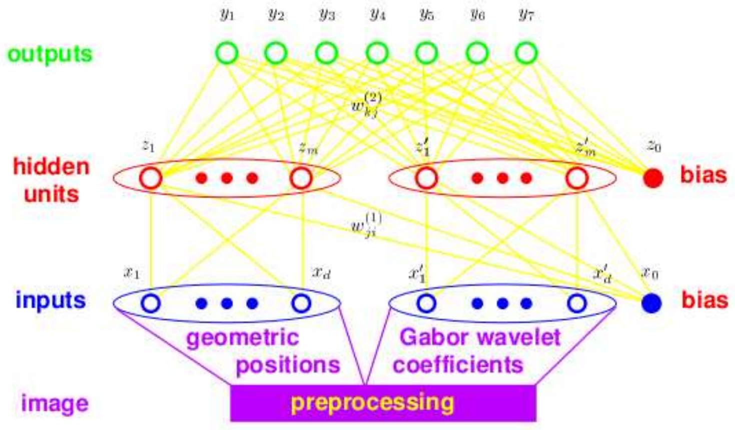


Figure 3.10: Zhang and Zhang 2-Layer Perceptron Architecture. The geometric component consists of the coordinates of 34 fiducial points from the image, while the Gabor wavelet component are the wavelet coeffients at the 34 pointsZhang and Zhang [83].

Gabor Wavelet Coefficients at Feature Points

Zhang and Zhang [83] perform experiments on a two-layer perceptron based recognition system. Zhang and Zhang filter images with Gabor filters and feeds them into a multi-layer perceptron varying the number of inputs and the number of hidden units. They also performs sensitivity analysis to determine which AUs are most relevant to facial expression recognition. Finally, they studied the significance of image scale and and makes conclusions as to which spatial resolution is best for FER.

The system used here combines holistic template matching and geometric feature-based systems. A set of feature points is located, and then Gabor wavelet coefficients are extracted from each point through image convolution. 213 images of female expressions are pre-processed, and 34 fiducial points are located manually. The image coordinates of these points form the first (geometric) component of the input. The second component of the input consists of the Gabor wavelet coefficients at these coordinates (See Figure 3.10). They use Gabor Kernels with 3 spatial frequencies and 6 orientations. In total, each image is represented by a vector of 612 (18 X 34) elements. Zhang performs "sensitivity analysis" by monitoring the change in recognition rate when various fiducial points are removed from the training process. The less the change, the less "useful" the point is. Finally they studies the significance of each spatial frequency by using a range of frequencies in all experiments. They comes to the following conclusions:

- Classification using geometric data alone does not exceed 70% for any number of hidden units where as when Gabor wavelet coefficients are included, or using Gabor wavelet coefficients alone, the recognition rate exceeds 80% for 4 on more hidden units.
- At least 2 hidden units are required for recognition rates greater than 50%.

- Between 5 and 7 hidden units are required for precise coding of facial expressions.
- Geometric data only improves recognition rates when hidden units are fewer than 5.
- The following 10 feature points have significantly less contribution to recognition rate: 16, 17, 18, 19, 20, 21, 23, 24, 30, 13. Points 3 and 29 fall along the median of the recognition rate. Zhang and Zhang observed that there was slight improvement in generalization after discarding the 12 points.
- Frequencies $k = \frac{\pi}{16}$ and $k = \frac{\pi}{8}$ consistently out performs $k = \frac{\pi}{32}$, $k = \frac{\pi}{4}$ and $k = \frac{\pi}{2}$ for all numbers of hidden units.

In conclusion, Zhang and Zhang performed several, comprehensive tests on the 2-layer perceptron architecture for FER. The most relevant conclusions are that FER is mainly a low frequency process and that the most useful fiducial features are around the eyes and mouth regions.

Feature Point Tracking and Bayesian Probability Theory

Cohn et al. [14] tracked feature points autonomously in regions of moderate to high texture over sequences of images. Each AU tested occurred at least 25 times in their image database, which consisted of photographs of 100 university students aged between 18 and 30 years.

Cohn et al. set up a controlled studio observation room with two cameras connected to one video recorder with a synchronized time-code generator. One of the cameras directly faced the subject, and the other was positioned 30 degrees to the right of the subject. Before normalization can take place, the medial canthus of both eyes and the uppermost point of the philtrum are manually marked in the initial image. The images are then automatically mapped using an affine transformation to a standard face model on which 37 features – 6 brow features, 8 eye features, 13 nose features, and 10 mouth features – are manually marked. See Figure 3.11. An optical flow algorithm is then used to automaticallty track the movement of these feature points through the image sequence.

Tracking three brow action units or action unit combinations (AU 1+2, AU 1+4 and AU 4), three in the eye region (AU 5, AU 6 and AU 7) and nine in the mouth region (AU 12, AU 25, AU 26, AU 27, AU 12+25, AU 20+25+16, AU 15+17, AU 17+23+24 and AU 9+17+25) their discriminant classifier recognition accuracy averaged at 91%, 88% and 81% respectively

3.3.4 Principal Component Analysis

PCA serves to represent a test image as a linear combination of base images (principal components). Depending on the size of the dataset that the components are derived from, the base images can be applied to new images by varying the contribution (eigenvalues) of each principal component (eigenvector). In PCA, each image in the dataset is projected

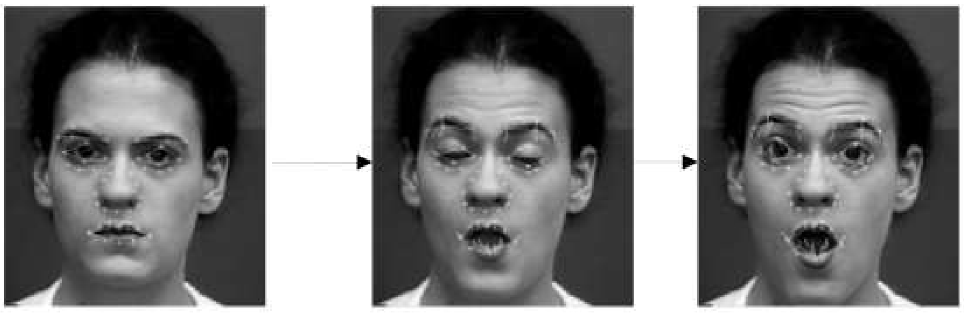


Figure 3.11: Concurrent Feature Point Tracking. Cohn et al. [14].

onto p principal components, where $p << n$, where n is the number of images. The greater the size of the training set, the better the eigenvectors will be able to generalize. This section looks at a novel method of *peeking*, which appears to improve generalization of the local PCA representation to match that of Gabor filter representations.

PCA Matches Gabor Filter Representations

Dailey and Cottrell [18] show that the local PCA representation yields quantitatively indistinguishable results from Gabor filter representations for the task of FER. They also show that linear discriminant analysis performed on the Gabor filter representation automatically locates the important regions that correspond to portrayed facial actions. They also introduce a method of *peeking* at unlabeled test sets to improve generalisation.

The Gabor filter representation is done using the standard method: Gabor filter jets are produced by convolving images with a bank of 40 filters (five scales by eight orientations), taking the magnitude of their complex-valued responses, and sub-sampling. PCA is then applied to the vector to reduce training time and improve classification generalization, and reduces the dimensionality of the vector from 41769 elements to 109 elements. For training, a nearest neighbor classifier is used. The classifier was tested on the full Gabor representation of the POFA[5] database, and correctly recognised 74.0% of images.

The Gabor jet representation is compared to various representations in which salient face regions are projected onto a set of 40 principal component eigenvectors from a large set of patches from randomly selected locations in the image database. Figure 3.12 shows the Gabor jet grid, the seven regions for local PCA used, the full grid for local PCA and the local PCA regions chosen by linear discriminant analysis.

Classification is performed using 13 standard back-propagation neural networks. Each network is trained using the images of 12 actors. The images of the 13th individual are used as a holdout set for early stopping, while the images of the 14th actor are used to test the the network's performance. The outputs of the 13 networks are then combined

[5]The Pictures Of Facial Affect database has 14 actors.

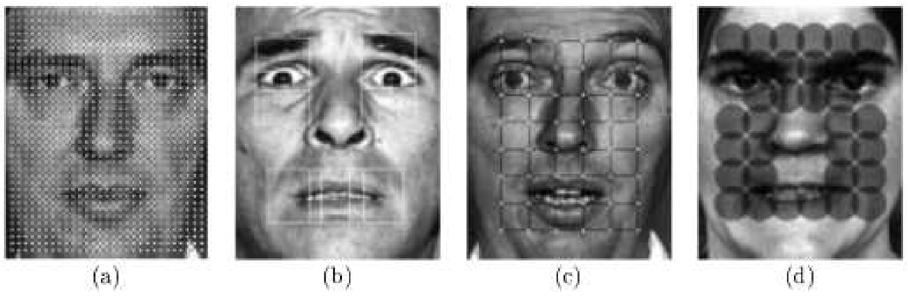


Figure 3.12: Image feature locations. (a) Gabor jet grid. (b) The seven regions for local PCA used by Dailey and Cottrell [18]. (c) Full grid for local PCA. (d) Local PCA regions chosen by linear discriminant analysis. Dailey and Cottrell [18].

using one of two ways. Online mode, in which the softmax[6] of each individual network's output vector is obtained and then averaged to obtain the aggregate response. In batch mode, the networks responses are calculated using the entire test set as well as their means and standard deviations of each output unit. When required to produce a response for an individual test item, instead of softmaxing the outputs, they are Z-scaled and averaged over the 13 networks. This peaking technique allows the classifier to consider the distribution of responses to the test set when producing the final response on a given test item.

The paper shows that their PCA based classifiers performed as well as Gabor representations. It is not clear how to choose one representation over the other. They do however conclude that there is a narrow "sweet spot" in classification performance when PCA is combined with Gabor jets.

3.3.5 Flow Fields

When computing optic flow, image intensity is modelled as a function of *x*, *y* as well as of time *t*. Optic flow is represented as velocity in the x and y directions: v_x and v_y, which are functions of the displacement in x and y over the change in time – $v_x = \frac{\Delta x}{\Delta t}$ and $v_y = \frac{\Delta y}{\Delta t}$. This section shows that flow fields yield excellent results:

- dense flow tracking with PCA outperforms both facial feature point tracking and high gradient component detection; and
- flow tracking works very well when the upper face is tracked separately from the lower face.

Feature Point Tracking, Dense Flow Tracking and High Gradient Component Detection

Lien et al. [51] put together a system that successfully recognises three upper face expressions using a combination of three approaches to extract facial expression information and train HMMs: (1) facial feature point tracking, (2) dense

[6]Softmaxing ensures all the output values are between 0 and 1 and sum to 1.

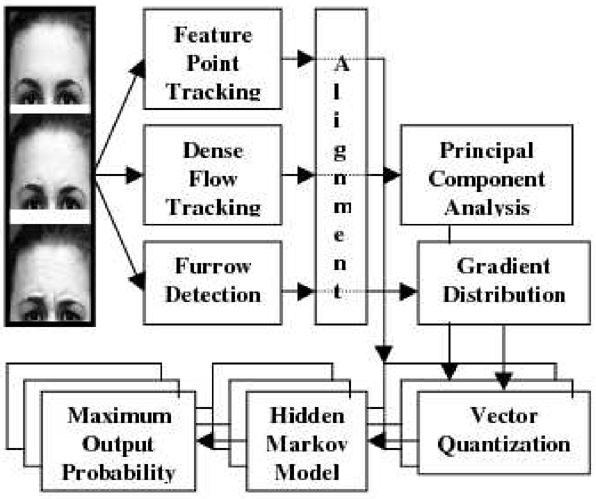


Figure 3.13: The system structure, Lien et al. [51].

flow tracking with PCA, and (3) high gradient component detection (i.e., furrow detection). Their best results are 95% and came from dense flow tracking, while gradient component detection and feature point tracking both yielded 85%. (See Figure 3.13.)

The data set consists of images of the upper half of the face and with the presence of brows in all samples these techniques are optimised for their data samples. The first two approaches are optical flow methods, used for their ability to extract texture information from facial skin and features. Facial feature point tracking is sensitive to subtle feature motion and Lien et al. use a 5-level pyramidal optical flow approach for its ability to detect large feature point movements (100-pixel displacement between two frames) and easily pickup eyelid movement. This approach however, loses information from areas not selected, such as the forehead, cheek and chin regions. Dense flow tracking can detect more detailed facial motion information and from larger regions of the face. Lien et al. track each individual pixel of the image using Wu's dense flow algorithm [80]. PCA can be used to compress the image data as well as to maintain the strong correlation between motion frames. Facial motion produces wrinkles and furrows in the skin and these come across as gray-value changes in the face image. High gradient component detection – their third method – is able to detect these gray-value changes. In training the HMM sets, the 12-dimensional displacement vectors from feature point tracking, the 20-dimensional weighted vectors produced by PCA, and the 26-dimensional mean and variance vectors from the high gradient component detection are vector quantized. Test sequences are evaluated by selecting the highest output probability from the HMM set.

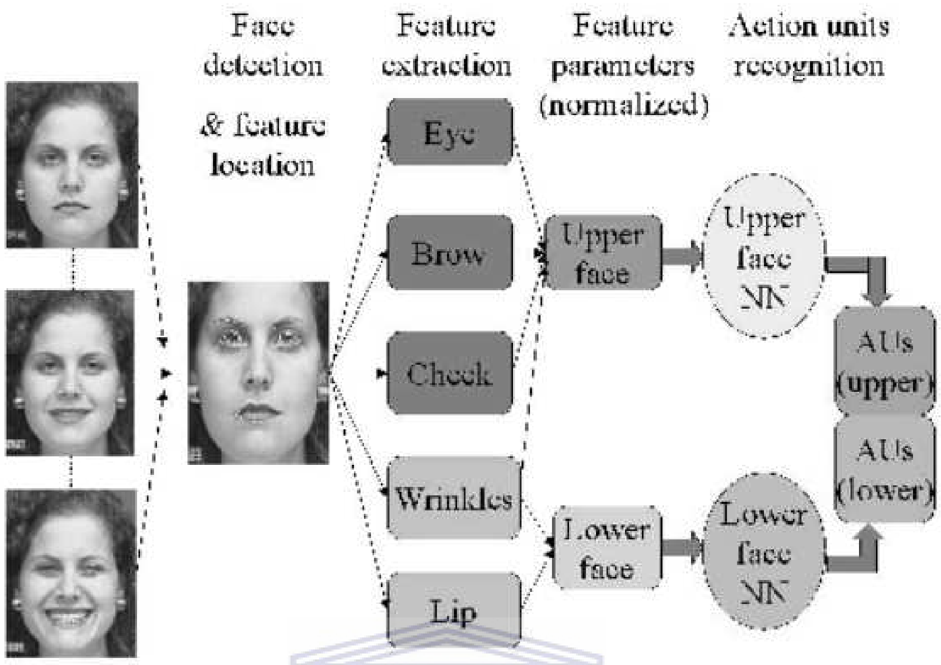


Figure 3.14: Recognizing Action Units for Facial Expression Analysis. li Tian et al. [50]

Multi-State Face and Facial Component Models

li Tian et al. [50] developed a system incorporating multi-state face and facial component models to track and model various fiducial features including eyes, brows, cheeks and lips. They performed flow analysis on the features and identified a set of details which they fed into a neural network. Their system attains a recognition rate of 96.7% for lower face AUs and 95% for upper face AUs, and recognizes AUs independently of each other.

The head orientation and face position are detected. As the subject performs a facial expression from neutral, subtle changes in the facial components are measured and represented as a set of feature parameters. In this system, there are two parameter groups for facial features: 15 parameters for the upper face describing eye shape, motion, state, brow and cheek motion and upper face furrows; and nine parameters for the lower face to describe the lip shape, lip motion, lip state and lower face furrows. The various representations are discussed in much more detail than is permitted here.

Seven basic upper face AUs (Neutral, AU 1, AU 2, AU 4, AU 5, AU 6, AU 7) and eleven basic lower face AUs (Neutral, AU 9, AU 10, AU 12, AU 15, AU 17, AU 20, AU 25, AU 26, AU 27, AU 23 + AU 24) are identified. Once the facial features are extracted and suitably represented, two neural networks are used to recognise the AUs – a 3-layer neural network with 15 inputs and 7 outputs for the upper face AUs, and a 3-layer neural network with 9 inputs and 11 outputs for lower face AUs.

Some of their most important conclusions are:

1. occlusion as a result of closing of the lips or blinking of the eyes degrades optical flow estimation;
2. both motion and feature appearance affect sensitivity to expression recognition;
3. facial feature measurements performed comparably with holistic analysis and Gabor wavelet representation;
4. 5 to 7 hidden units are needed to code 7 upper face AUs individually, and 10 to 16 hidden units when they occur either singly or in complex combinations.

3.3.6 Holistic Spatial Analysis and Template Matching

Holistic spatial analysis is an appearance-based method for analysing images. Because using all the pixels in an image could quickly become too resource intensive, holistic spatial analysis is often used in conjunction with some form of dimensionality reduction, such as PCA. Template matching seeks to impose templates of the region of interest on an image. Templates can be predefined – based on edges or regions – or deformable – based on facial contours. Templates are hand-coded and statistical correlation is used to locate features. In combination, template matching and holistic spatial analysis can be used to track and locate fiducial features, as well as for dimensionality reduction. This section shows that:

- holistic spatial analysis performs better than feature point tracking, and flow fields, but the combination of the three performed the best;
- the discrete SNoW-NB classifier outperforms SNoW, SNoW-NB, NB-Gaussian, NB-Cauchy and TAN classifiers; and
- PCA+LDA works well for dimensionality reduction in model based tracking

Holistic Spatial Analysis using PCA

Bartlett et al. [3] performed experiments with three techniques: holistic spatial analysis, explicit measurement of features and estimation of motion flow fields. They investigated the three techniques individually before combining them to form a hybrid system.

With holistic spatial analysis, a 2 layer neural network with 10 hidden units, and 1 output unit for each of the 6 actions was used on principal components of gray level images. The pixel intensities of the difference images are concatenated to form a vector, which then became the input for their network. The activities are calculated as the sequential weighted sum of their inputs passed through a sigmoidal hyperbolic transfer function. Their neural network outputted a 1 at one unit and 0's everywhere else when the appropriate action was detected.

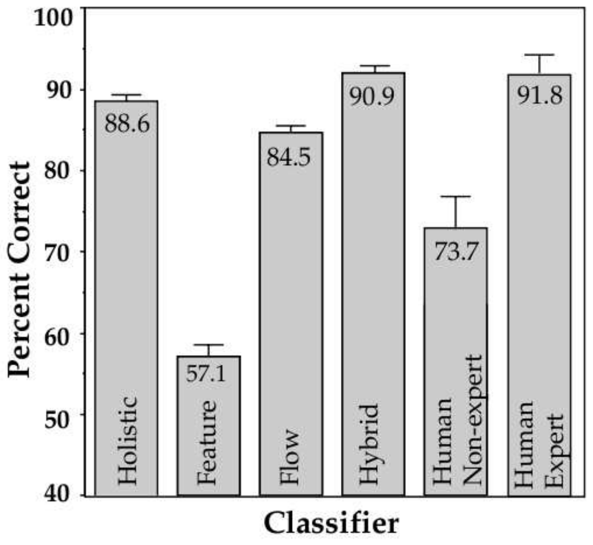


Figure 3.15: Summary of results, Bartlett et al. [3].

Their system enabled them to classify 6 upper facial actions with 90.1% accuracy – as good as highly trained experts on FACS coding. See Fig 3.15.

They used video sequences of 20 subjects, each performing a mean of 4 actions. Image registration was performed in which the faces are aligned, cropped and scaled to center pupil location. Their best performance of 92% was for classifying 6 actions and was achieved by combining the three techniques.

Automatic Segmentation and Recognition using a HMM Layer and Markov Model Layer Hybrid

Cohen et al. [13] propose a new architecture of HMMs for automatically segmenting and recognizing expressions from videos using a multi-level architecture composed of both a HMM layer and a Markov model-based system.

The face tracking used was originally developed by Tao and Huang [70] and is called the Piecewise Bezier Volume Deformation (PBVD) tracker. A 3D wire frame model of the face is constructed upon which the generic face model is warped to fit selected facial features. Using 16 surface patches embedded in Bezier volumes, the model is guaranteed to be continuous and smooth. Using template matching, 2D image motions are modeled as projections of the true 3D motions onto the image plane. The 3D motion can be estimated using 2D motions of many points on the mesh.

Recovered motions are represented in terms of magnitudes of some predefined motion of various facial features. Thus, each feature motion corresponds to a simple deformation on the face. Facial expressions are modelled as a linear combination of these deformations. Cohen et al. [13] performed three sets of experiments. The first two used the assumption of strong independence between features, while the last one relaxed that constraint.

The first set of experiments use discretized features using the *Sparse Network of Winnows* (SNoW) and *SNoW-*

Naive-Bayes (SNoW-NB) classifiers. SNoW was developed by [1] and has been applied successfully to natural language processing for context-sensitive spelling correction [66] and face detection in images [37]. Benefits of using discrete features are that no assumptions are made on the distribution of features, and because SNoW is not a fully connected network like multi-layered perceptrons, not as many training examples are needed. One problem is that complexity increases exponentially with the number of values that features can take on. The SNoW-NB classifier is a probabilistic classifier in which the features are assumed to be independent of class.

The second set of experiments is performed on Cauchy Naive Bayes classifiers - Continuous Naive Bayes with a Cauchy distribution. This classifier is a Naive Bayes classifier that uses the Cauchy distribution in place of the Gaussian distribution for modelling the probability of features given a class label. Cohen et al. [13] also propose an innovative algorithm for determining the best distribution to use.

Since strong independence is highly unlikely in the task of FER, it becomes necessary to find a structure that better captures the dependencies among features; this leads to Tree-Augmented-Naive Bayes (TAN) classifiers. TAN classifiers were introduced by Friedman et al. [33], and are represented as Bayesian networks. TAN models are more complicated than Naive-Bayes models but are not fully connected. The TAN model is calculated by determining the TAN structure that maximizes the likelihood function given the training data. Cohen et al. [13] employed the Chow-Liu algorithm [12] for constructing the TAN networks.

The third group of experiments exploit the temporal information in the video using multi-level HMMs to discriminate different expressions. They propose an innovative method that automatically segments the video to the different facial expression sequences using a multi-level HMM structure. The first level consists of several independent HMMs each corresponding to a different emotion. The state sequence of the HMMs then becomes the input for another higher level Markov model. This not only segments the video sequence, but it also increases the discrimination between the classes by finding both the probability of each sequence displaying one emotion, and that of not displaying all the other emotions. Their paper has many more details of the multi-level HMM system. Figure 3.16 shows the Multi-Level HMM architecture. See Figure 3.17 below for a table of results.

As can be seen in the table in Figure 3.17, the discrete SNoW-NB outperforms all of the classifiers. The TAN classifier comes in second, followed by NB-Chauchy, NB-Gaussian and SNoW. Cohen et al. [13] goes on to perform person-independent experiments. Each classifier was trained using images from $N-1$ subjects, and then tested on the Nth subject alone. In this case, TAN provides the best results. It was also shown that the Cauchy distribution provides better results than the Gaussian distribution assumption.

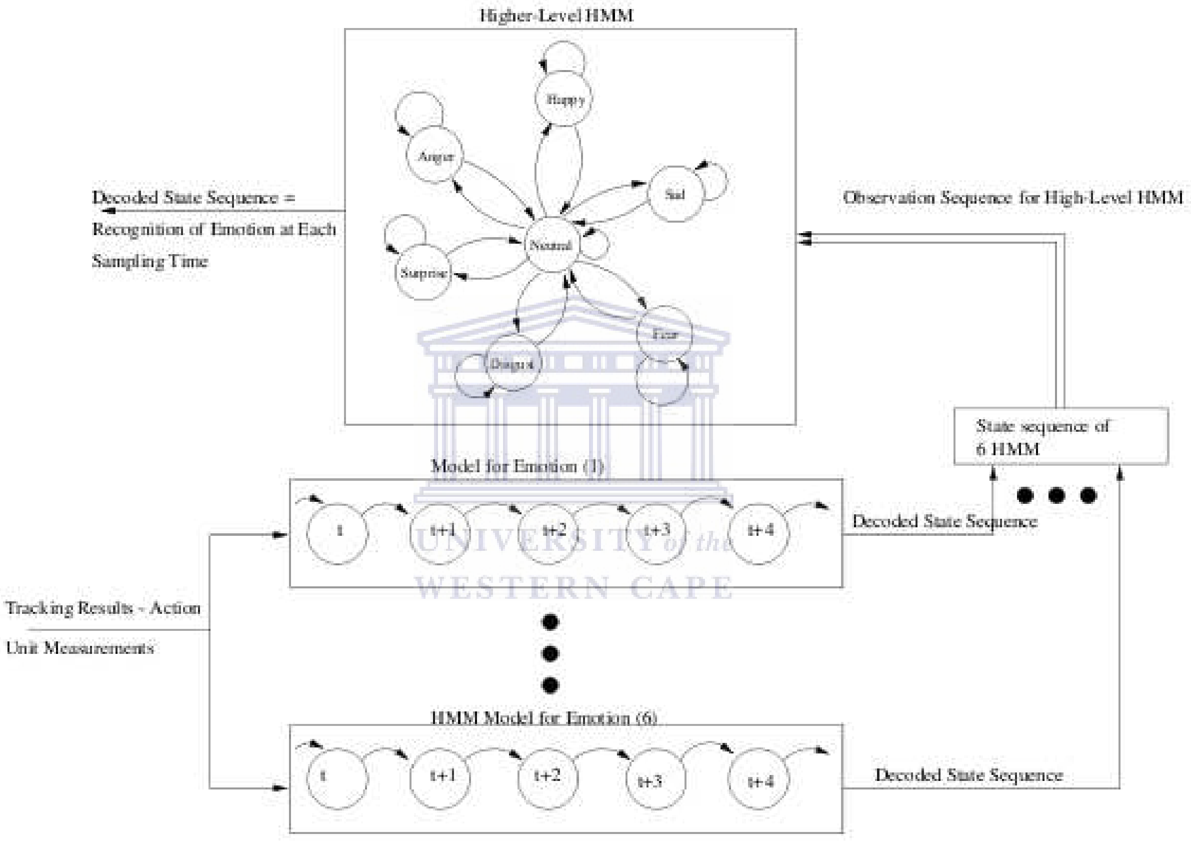
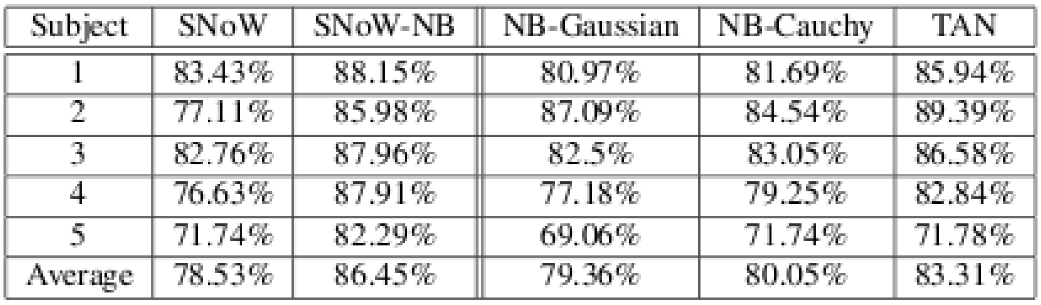
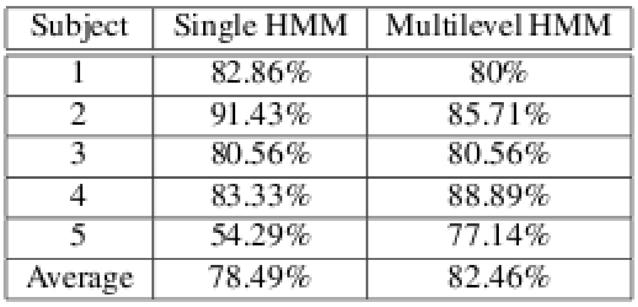

Higher-Level HMM
Happy
Anger
Sad
Neutral
Surprise
Fear
Disgust
Decoded State Sequence = Recognition of Emotion at Each Sampling Time
Observation Sequence for High-Level HMM
State sequence of 6 HMM
Model for Emotion (1)
t
t+1
t+2
t+3
t+4
Decoded State Sequence
Tracking Results - Action Unit Measurements
HMM Model for Emotion (6)
Decoded State Sequence

Figure 3.16: Multi-Level HMM architecture for automatic segmentation and recognition of emotion from Cohen et al. [13]



Subject	SNoW	SNoW-NB	NB-Gaussian	NB-Cauchy	TAN
1	83.43%	88.15%	80.97%	81.69%	85.94%
2	77.11%	85.98%	87.09%	84.54%	89.39%
3	82.76%	87.96%	82.5%	83.05%	86.58%
4	76.63%	87.91%	77.18%	79.25%	82.84%
5	71.74%	82.29%	69.06%	71.74%	71.78%
Average	78.53%	86.45%	79.36%	80.05%	83.31%



Subject	Single HMM	Multilevel HMM
1	82.86%	80%
2	91.43%	85.71%
3	80.56%	80.56%
4	83.33%	88.89%
5	54.29%	77.14%
Average	78.49%	82.46%

Figure 3.17: Top: Person-dependent facial expression recognition accuracies using frame based methods. Bottom: Results from the multi-level HMM architecture for automatic segmentation and recognition of emotion from Cohen et al. [13].

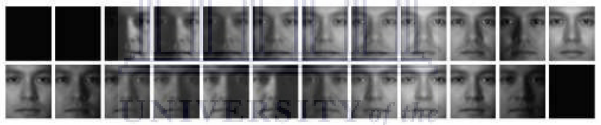


Figure 3.18: Different Angles of Illumination, Li et al. [49].

Real-Time Model Based Tracking with PCA+LDA Dimensionality Reduction

Li et al. [49] develop a system that tracks the rigid motion of a front-facing human face while dealing with illumination changes in real time. A model-based procedure is used for tracking. This approach achieves 86% recognition success.

Their user-independent appearance model is fitted to an incoming image using an efficient and robust fitting algorithm [8]. Facial expression recognition is also performed using a model based approach. A model is constructed with n parameters so that faces are represented by a point in an n-dimensional space of deformations. Images representing similar expressions are mapped to the same general area. A probabilistic procedure is then used to combine image information with that in the manifold information to estimate a posterior probability for each expression.

A generic facial appearance model is trained on the PIE [69] database. Each illumination is averaged across all identities resulting in an average image for each illumination direction (see Fig 3.18). Target images are fitted to the model by estimating the motion and appearance – parameters which minimize an error function. Their paper contains

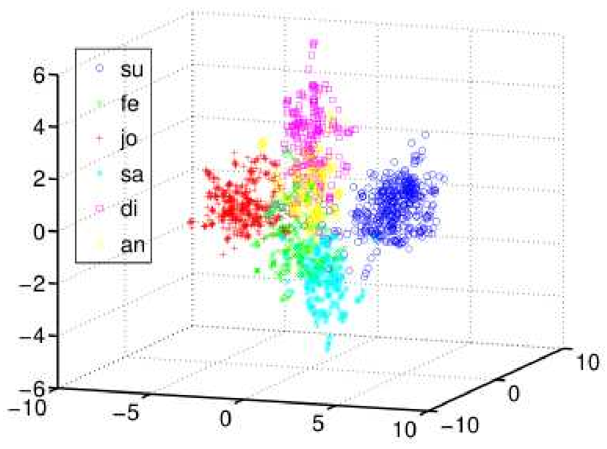


Figure 3.19: Manifests. Li et al. [49].

more technical details of this process.

Once motion and illumination parameters have been estimated, an illumination normalized version of the rectified image is computed. The high dimension of illumination-normalized images is then reduced using PCA+LDA. The images are processed using PCA and 90 eigenvectors are retained. LDA is then applied, producing a 5-dimensional subspace. Their model of a prototypic facial expression is the manifold that contains the set of trajectories of that expression in the database. The complete facial expression model is the union of the six manifolds associated with each prototypic expression.

3.3.7 Independent Component Analysis (ICA)

In PCA, the principal vectors are uncorrelated, but they are not necessarily statistically independent. ICA is a stricter form of PCA, in which the principal vectors are statistically independent as well as uncorrelated. Under ICA, the basis vectors are known as independent components and the projections from the dataset onto each independent component is statistically independent of all other projections. This section shows that ICA very closely matches Gabor wavelets in performance.

ICA Matches Gabor Wavelets in FER

Donato et al. [20] explores and compares a range of approaches to face image representation for the purpose of classifying Action Units (based on FACS) in videos. Their techniques included holistic spatial analysis such as PCA, ICA, local feature analysis (LFA) and LDA; analysis of facial motion through estimation of optical flow; and methods based on the outputs of local filters, such as Gabor wavelet representations and local principal components. Their highest recognition accuracy was 96% and was obtained for both Gabor wavelets and for ICA. Interestingly, both

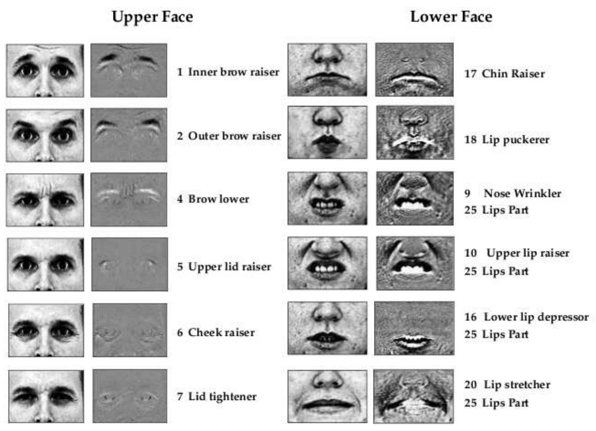


Figure 3.20: Comparing Facial Actions: The 12 Action Units Classified, Donato et al. [20].

techniques are gray-level texture filter methods.

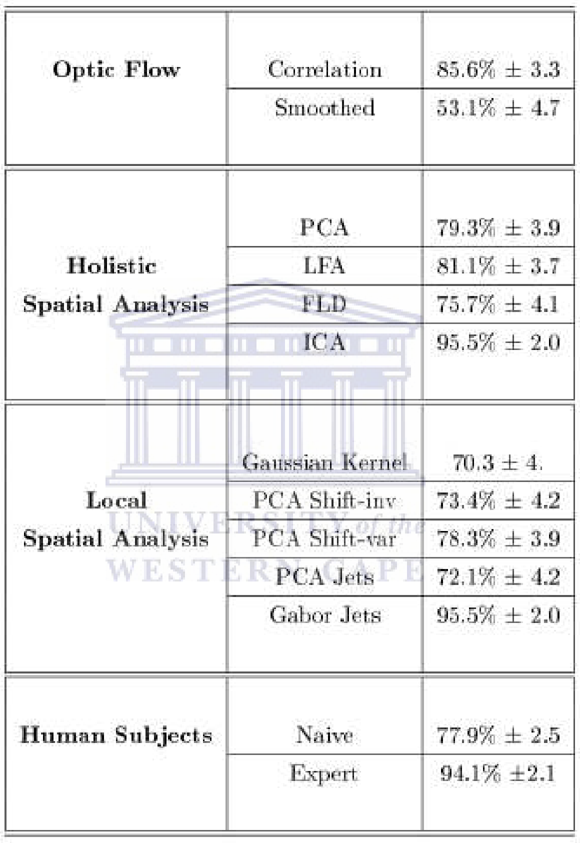
Their image dataset contained data from 20 subjects, each performing 12 actions: 6 upper face actions and 6 lower face actions. There are a total of 111 action sequences, which could be divided into upper and lower-face categories because of the lack of influence of motion from the opposite half [23].

The experiments performed by Donato et al. [20] show that local spatial filters are important for analysis of facial expressions. Both Gabor filters and ICA, while applied to images as a whole, produced localized basis images. However, spatial locality alone is insufficient for good classification. Results showed that high spatial frequencies are more important than low spatial frequencies for classifying facial actions. Regularization of image representations, such as smoothing and quantizing performed during optic flow analysis, degrades performance by as much as 30%. This suggests that optic flow analysis is not suitable for the complex task of facial action classification. They suggest that higher frequencies are necessary when a more detailed level of analysis is required. Furthermore, ICA and Gabor filters both reduce redundancy in image representations. See Figure 3.21 for more results.

3.3.8 Linear Discriminant Analysis (LDA)

LDA is closely related to PCA in that they both look for linear combinations of variables which best explain the dataset. While PCA ignores any difference in class, LDA attempts to model the differences between the classes of data. This section shows that:

- local Gabor filters with PCA and LDA outperforms PCA alone; and
- *pyramid Gabor features* outperform traditional Gabor features and involve fewer computations.



Optic Flow	Correlation	85.6% ± 3.3
	Smoothed	53.1% ± 4.7
Holistic Spatial Analysis	PCA	79.3% ± 3.9
	LFA	81.1% ± 3.7
	FLD	75.7% ± 4.1
	ICA	95.5% ± 2.0
Local Spatial Analysis	Gaussian Kernel	70.3 ± 4.
	PCA Shift-inv	73.4% ± 4.2
	PCA Shift-var	78.3% ± 3.9
	PCA Jets	72.1% ± 4.2
	Gabor Jets	95.5% ± 2.0
Human Subjects	Naive	77.9% ± 2.5
	Expert	94.1% ±2.1

Figure 3.21: Comparing Facial Actions: Table of Results, Donato et al. [20].

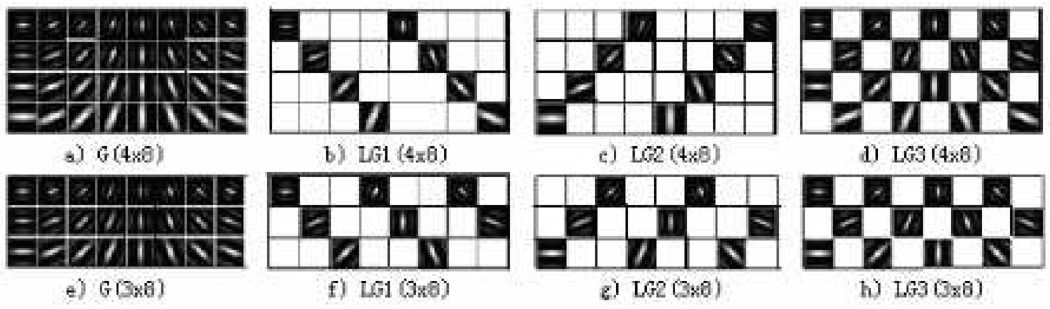


Figure 3.22: The PCA+LDA Filters Used by Deng et al. [19].

PCA with LDA Outperforms PCA alone

Deng et al. [19] perform experiments with PCA, LDA, PCA + LDA, and Gabor filters. Their results show that using PCA and LDA in combination yields the best results and achieves both dimension reduction and good recognition performance compared to the traditional 5X8 filter bank.

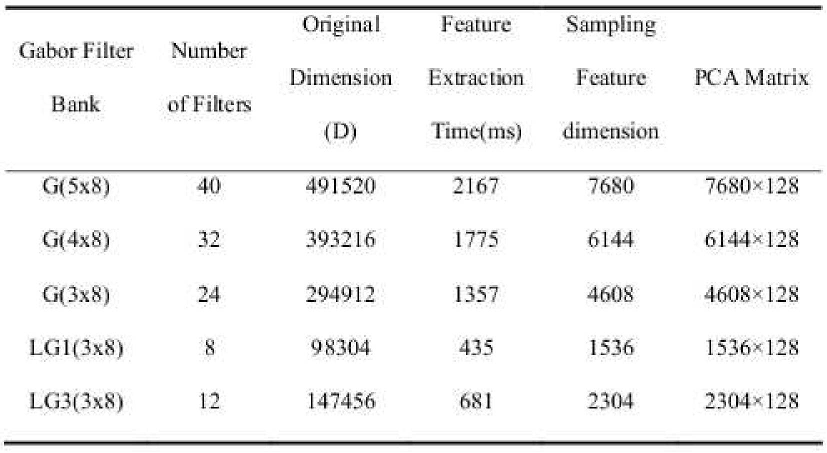
Deng et al. argue that the standard Gabor filter bank of 5 frequencies and 8 orientations produces a feature vector that is redundant and correlative. Their new filter bank is a subset of the 40 filters and either selects only one frequency for each orientation or increases the interval between the neighboring frequencies with the same orientation.

Figure 3.22 shows examples of several global and local Gabor filter banks. The classical algorithm for Gabor filter application is to convolve an image with a bank of Gabor filters, concatenate the resulting filtered images, subsample and then normalize to unit length. This produces a Gabor jet. If the number of images in the bank is reduced – as shown by the local filter banks in Figure 3.22.b,c,d,f,g,h – then the dimensions of the Gabor jet will be reduced.

Deng et al. also compare two methods for reducing dimensionality by finding linear combining features: PCA and LDA. PCA seeks a projection that best *represents* the original data in a least-squares sense, while LDA seeks a projection that best *separates* the data in a least-squares sense. For a set of N sample images represented by a t-dimensional Gabor feature vector, PCA can be used to find a linear transformation mapping the original t-dimensional feature space into an f-dimensional feature subspace. A limitation of PCA is that important information for discrimination between different classes may be lost. LDA serves to maximize the between-class scatter while minimizing the within-class scatter.

Their conclusions are:

1. local Gabor filter banks out performs global Gabor filter banks in terms of time taken, dimensionality, computation resources and storage (see Figure 3.23), and even performance in some situations;
2. while PCA significantly reduces dimensionality, it may lose important information for discrimination between classes;



Gabor Filter Bank	Number of Filters	Original Dimension (D)	Feature Extraction Time(ms)	Sampling Feature dimension	PCA Matrix
G(5x8)	40	491520	2167	7680	7680×128
G(4x8)	32	393216	1775	6144	6144×128
G(3x8)	24	294912	1357	4608	4608×128
LG1(3x8)	8	98304	435	1536	1536×128
LG3(3x8)	12	147456	681	2304	2304×128

Figure 3.23: Preprocessing time and dimensions for local and global Gabor filter banks, Deng et al. [19].

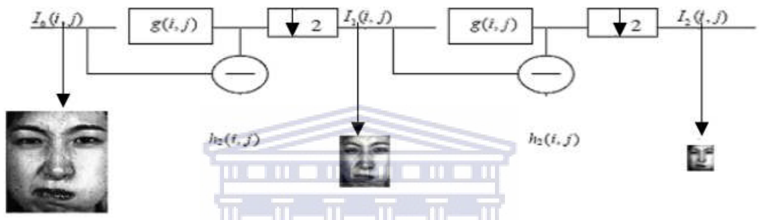


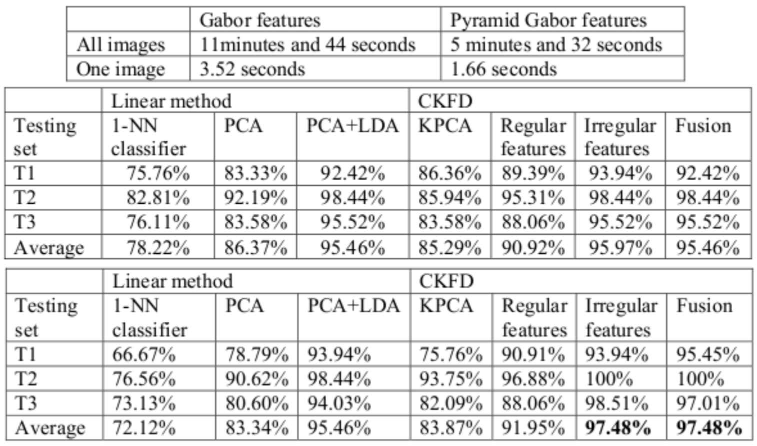
Figure 3.24: Decomposition of I_0 into The Lower Resolution Images I_1 and I_2, Yang et al. [81].

3. PCA with LDA outperforms PCA in terms of recognition, and the dimensionality is reduced significantly; and
4. discarding no more than the first three principal components will produce the best performance rates.

Their best performance was 97.33% correctly classified images and came from using local Gabor filter bank, with 3 frequencies and 8 orientations, and PCA with LDA and eliminating the first two principal components.

Pyramid Gabor Features and Complete Kernel Fisher Linear Discriminant Analysis

Yang et al. [81] present an approach that uses pyramid Gabor features and complete kernel Fisher linear discriminant analysis (CKFD) . Their experiments show that Pyramid Gabor features outperform traditional Gabor features while at the same time involving much fewer computations. Yang et al. also apply the CKFD to FER and find that the irregular subspace is more discriminant than the regular information subspace. Their proposed pyramid algorithm decomposes images into different low resolution images using a Laplacian pyramid. Figure 3.24 shows a series of decompositions using a pyramidal algorithm. Where classical Gabor filters only extract the Gabor feature centering frequency k and orientation o. Yang, et al's proposed pyramid Gabor feature extraction algorithm first decomposes the input image into different channels. Then, for every low frequency channel, Gabor features are extracted using a set of Gabor filters



	Gabor features	Pyramid Gabor features
All images	11minutes and 44 seconds	5 minutes and 32 seconds
One image	3.52 seconds	1.66 seconds

	Linear method			CKFD			
Testing set	1-NN classifier	PCA	PCA+LDA	KPCA	Regular features	Irregular features	Fusion
T1	75.76%	83.33%	92.42%	86.36%	89.39%	93.94%	92.42%
T2	82.81%	92.19%	98.44%	85.94%	95.31%	98.44%	98.44%
T3	76.11%	83.58%	95.52%	83.58%	88.06%	95.52%	95.52%
Average	78.22%	86.37%	95.46%	85.29%	90.92%	95.97%	95.46%

	Linear method			CKFD			
Testing set	1-NN classifier	PCA	PCA+LDA	KPCA	Regular features	Irregular features	Fusion
T1	66.67%	78.79%	93.94%	75.76%	90.91%	93.94%	95.45%
T2	76.56%	90.62%	98.44%	93.75%	96.88%	100%	100%
T3	73.13%	80.60%	94.03%	82.09%	88.06%	98.51%	97.01%
Average	72.12%	83.34%	95.46%	83.87%	91.95%	**97.48%**	**97.48%**

Figure 3.25: Pyramid Gabor Features: CKFD vs PCA + LDA. Yang et al. [81].

with a single frequency but different orientation. Finally, sub-sampling is performed followed by normalization and concatenation – the same procedure followed in creating a Gabor jet. Since I_1 and I_2 are of a lower resolution than I_0, it is clear that applying Gabor filters to them requires fewer computations than traditional Gabors.

Yang et al. introduce the CKFD algorithm. PCA and LDA have often been used in reducing the dimensionality of features for the task of FER. Kernel PCA [46] comes from problems that have a complex distribution – non-linear problems can be solved using non-linear methods. KPCA is PCA with an undefined non-linear kernel.

Figure 3.25 shows some experimental results: the topmost table shows that pyramid Gabor features are twice as fast as standard Gabor features, while the middle and lower two tables show that the standard Gabor features-based method is outperformed by the pyramid Gabor method, respectively.

One important conclusion is that irregular information is more discriminant for the task of FER than regular information.

3.3.9 Point Distribution Models (PDM)

Active shape models (ASMs) [15] use information from points around *landmarks* on the face to compute a PDM and an image intensity profile. The landmark points around an image are represented using their coordinates in a single vector, or *shape*. In the training stage, collected shapes are aligned to the same coordinate frame after which PCA is applied to compute the *mean shape*. Any shape can then be approximated using the mean shape $\overline{\mathbf{x}}$, a scaled eigenvector matrix $\mathbf{P}$ and b_i, which defines the shape parameters for the i^{th} shape,

$$\mathbf{x}_i = \overline{\mathbf{x}} + \mathbf{P}b_i,\ b_i = \mathbf{P}^T(\mathbf{x}_i - \overline{\mathbf{x}}). \tag{3.1}$$

Dimensionality is then reduced further by selecting only the eigenvectors that correspond to the largest eigenvalues and adding an error function to account for the removed eigenvectors:

$$\|b_i\| \leq \beta\sqrt{\lambda_i},\ 1 < i < M, \tag{3.2}$$

where β is a constant usually from 1-3, λ_i is the i^{th} eigenvalue and M is the total number of the selected vectors. This is done to ensure that only the allowable shapes are represented by Eq. 3.2

A statistical gray-level structure model is then formed by investigating profiles perpendicular to the shape boundary for each point. Computing intensity derivatives and normalizing them ensures some tolerance to global intensity changes. First, the mean shape is initialized. In a direction perpendicular to the shape profile, each point is moved outwards in an attempt to identify its correct position. Minimizing the Mahalanobis distance gives the displacement for each landmark point between the training and test models. The shape parameters are updated and the procedure repeated for all points until convergence occurs.

This section shows that:

- a PDM, Gabor Filters and ANNs strategy does not perform as well as Gabor or ICA.

A Region-Based Approach using PDM, Gabor Filters and ANNs

Koutlas and Fotiadis [43] investigate the use of a PDM, Gabor filters and artificial neural networks in FER. They show that a region-based approach can be used successfully and create a system that recognizes the 7 prototypic expressions automatically. Region-based approaches are useful in removing unnecessary information and artefacts from the FER process.

Their system automatically locates 74 landmark points using active shape models, from which 20 fiducial points are derived. A feature vector is formed from a region around each landmark point. The size of this region is parameterized and used in control experiments to determine an optimal size. The feature vector is then filtered via a bank of Gabor filters before being fed into a feed-forward artificial neural network.

Koutlas and Fotiadis opt to use 18 Gabor filters with three scales and six orientations. Koutlas and Fotiadis showed that by selecting certain scaling parameters, filters can be ensured against overlapping over image surface. They exploit this technique in an attempt to reduce redundant information.

A feed forward back propagation ANN is used. The mean square error function is used in training this ANN and the number of epochs is 500. Ten-fold cross validation is used. 7 experiments are conducted in which pixel regions are gradually increased in size. 1x1, 3x3, 5x5, 7x7, 9x9, 11x11, 13x13 pixel size regions are tested, and experimentally, 9x9 showed to have the best results (90.2%). The gradual increase in accuracy as the region gets larger shows that it uses more information that better describes facial geometry.

Their set of 20 fiducial points decreases the dimensionality of the feature vector by approximately 40% without any loss in accuracy. The pixel based approach was modified to accommodate information from neighboring pixels, called regions. This is especially useful in reducing errors that occur from imprecise identification when detecting features automatically. The method presented here achieves a maximum of 90.2% accuracy on the JAFFE database but does not perform well when trying to differentiate between sadness and fear.

3.3.10 Complete Systems

Much of the research presented thus far deals with the recognition of facial expressions in controlled environments where subjects have their heads still with very little in-plane rotation and hardly any out-of-plane rotation at all. Subjects either look directly into a camera or look at a predetermined location and have a camera directed at their face to capture the expressions made. This limits the practical application of those systems severely. This section presents a brief look at 3D warping of images into canonical views, and a combination of holistic template matching and a geometric feature based system that uses Gabor wavelet coefficients.

A Complete System using Gabor Filtered Images, SVMs and HMMs

Bartlett et al. [4] explore an approach based on the 3D warping of images into canonical views. Their 3D model is used as a front end for a system employing SVMs and HMMs. They create *deformable 3D face models* by fitting 3D face models to the image plane, texturing those models using the original image frame, rotating the model to frontal views, warping it to a canonical face geometry and finally rendering the model back into the image plane.

3D pose estimation was done by mapping landmark positions in the image plane to a canonical wire-mesh face model. The model was moulded to the the subject's face using an iterative least squares triangulation approach, in which camera parameters and 3D coordinates of the 8 features are estimated. A scattered data interpolation technique [64] then modifies the 3D model so that it fits the 8 feature points. A stochastic particle filtering approach [4] is then used to estimate the most likely rotation and translation parameters of the 3D model. In this manner images of faces with out-of-plane rotation are rotated back to face the camera.

The dataset used in their experiments was of spontaneous facial expressions from 17 freely behaving individuals. There are 3 Asian subjects, 3 African American and 11 Caucasians, of which 3 wore glasses. The images exhibited out-of-plane head rotation of up to 75 degrees. The Action Units classified in their experiments are AU 45 (blinks), AU 1+2 (brow raise) and AU 4 (brow lower), and there are at least 14 examples of each AU.

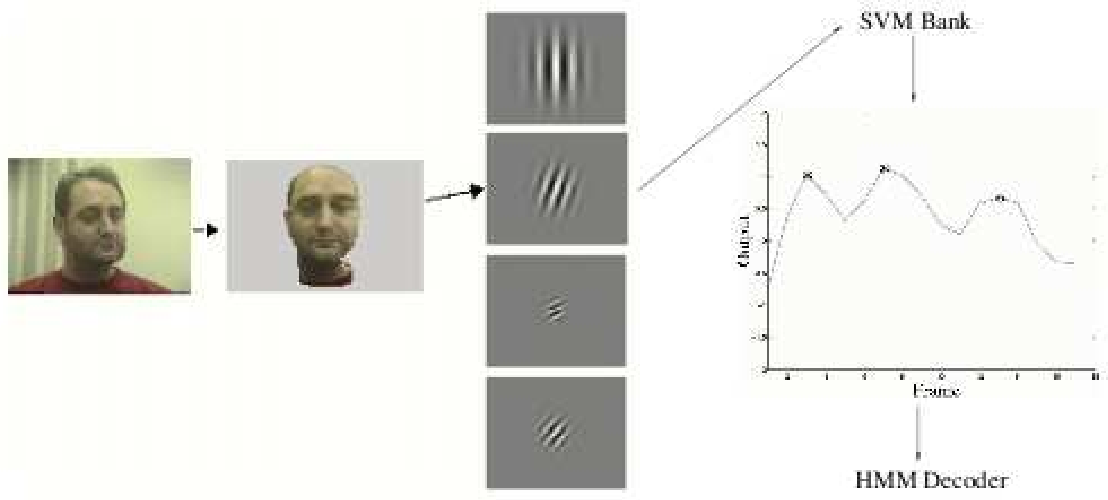


Figure 3.26: The Flow Diagram of the Recognition System, Bartlett et al. [4].

From the rotated facial images, eye and brow subregions are cropped and histogram equalization is performed on them. The equalized images are then convolved with a bank of Gabor filters at 5 spatial frequencies and 8 orientations, normalized to unit length and down-sampled by a factor of 4 before being channelled into a bank of non-linear SVMs. Generalization was tested using leave-one-out cross-validation. The trajectories of SVM outputs are then channelled to HMMs, thus training them to classify facial actions without knowing which frame contained the action peak.

Bartlett et al. succeeded in attaining a classification rate of 95.9% using SVMs to discriminate between frames of blinks versus images of non-blinks. A mixture of Gaussians model was used. Their HMMs attained a best performance of 98.2% using 6 hidden states and 7 Gaussians. Their final experiment attempted to discriminate between the three action units localized around the eyebrows. They used 3 binary SVMs, and fed the output into an HMM for classification. The best accuracy attained was 78.2% using 10 states and 7 Gaussians.

Real-Time Facial Action Analysis using PCA

Kapoor and Picard [42] created a fully automatic framework to analyse the facial actions and head gestures in real-time. They first robustly track the pupils. Pupil locations are used to localize regions of interest – namely the eyes and eyebrows – from which shapes are statistically recovered. The shape parameters are used as input to SVMs. HMMs that use pupil positions in consecutive frames as observations are used to detect head gestures. Frames of head gestures are successfully detected 78.46% of the time, recognition accuracy for each individual AU is 67.83% and that of AU combinations is 61.25%. Figure 3.27 shows an overview of the system.

Kapoor uses an infrared sensitive camera equipped with infrared LEDs to highlight and track pupils. The infrared

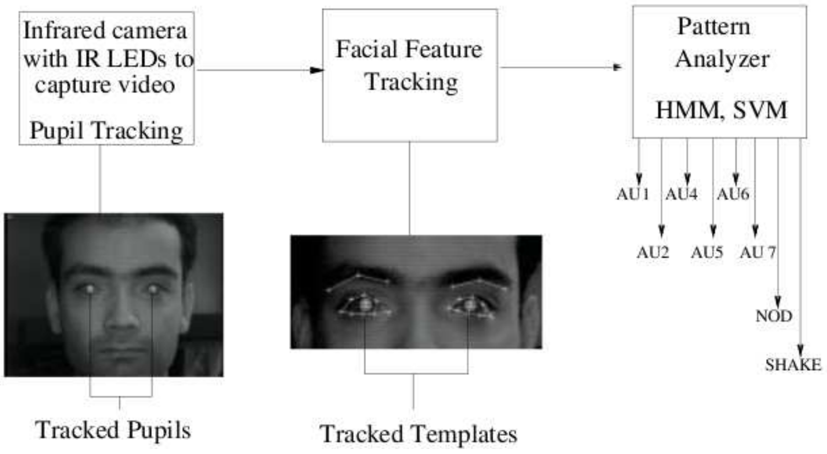


Figure 3.27: Kapoor Overall, Kapoor and Picard [42].

camera has two concentric rings of infrared LEDs. The first is on the optical axis and produces the red-eye effect, while the second is off axis and keeps the scene at the same illumination. Two images are thus produced, which are interlaced into a single frame. A difference image is then created and used to track the pupils. The performance of such a tracker is reportedly efficient, accurate and consistent. However, the specularity of objects like glasses and earrings allow them to be misinterpreted as pupils. Interlacing and motion artifacts also cause problems in detecting pupil locations.

The system then employs template matching to find and track facial features surrounding the pupils. Such a template is shown in Figure 3.28. [42] discusses and experiments with three methods for recovering shape parameters. The first – principal component analysis – assumes a linear relationship between shape parameters and image observations. The second discusses the same problem using a more general Bayesian estimation framework, while the last addresses *belief propagation* as a solution to the problems of the first two. [42] presents more details in his thesis paper. Because PCA is the only method implemented and worked in real-time, more emphasis is placed on its results. However, [42] does show that belief propagation does not provide significantly better results compared to the PCA approach: the average RMS (root mean square) per control point location for PCA was 0.715, while for belief propagation it was 0.815.

Figure 3.29 shows leave-one-out recognition results for action unit combinations. SVMs are trained for each facial action unit. The videos have a lot of occlusion and head movements, which lowers recognition rates. This explains the poor comparison to pre-processed or manually normalized image sets. The average recognition rate for each individual AU was 67.83% while that for AU combinations was 61.25%.

The pattern analyzer consists of two HMMs, one for head nods and the other for head shakes. 5 observation symbols which correspond to the head moving up, down, left, right or none are used as inputs for both classifiers.

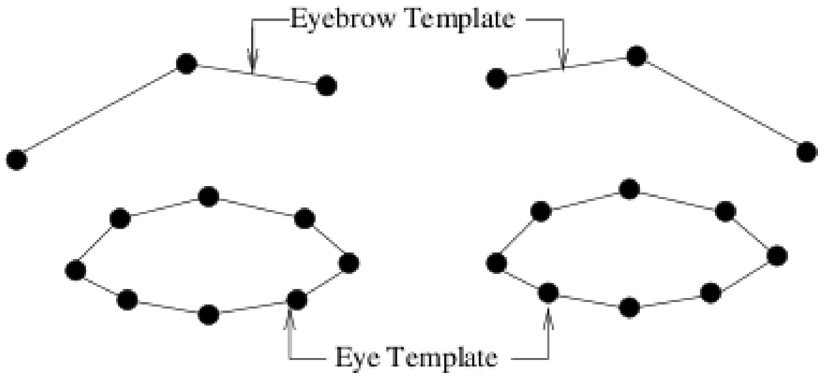
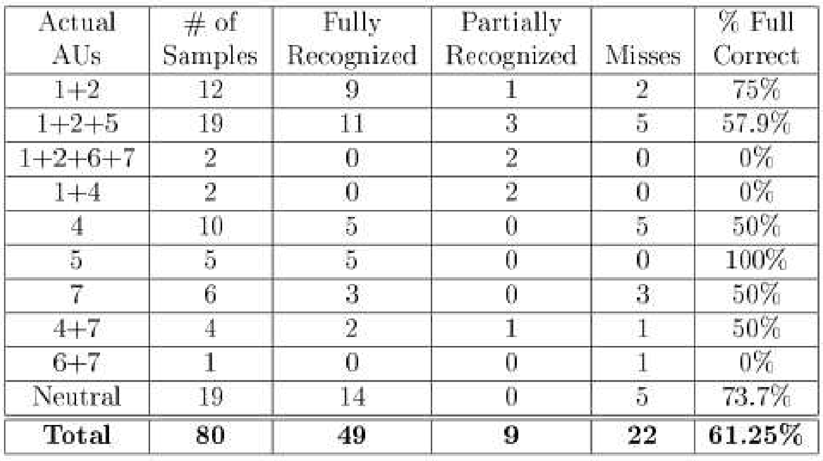


Figure 3.28: Eye and Eyebrow shape templates, Kapoor and Picard [42].



Actual AUs	# of Samples	Fully Recognized	Partially Recognized	Misses	% Full Correct
1+2	12	9	1	2	75%
1+2+5	19	11	3	5	57.9%
1+2+6+7	2	0	2	0	0%
1+4	2	0	2	0	0%
4	10	5	0	5	50%
5	5	5	0	0	100%
7	6	3	0	3	50%
4+7	4	2	1	1	50%
6+7	1	0	0	1	0%
Neutral	19	14	0	5	73.7%
Total	**80**	**49**	**9**	**22**	**61.25%**

Figure 3.29: Kapoor Full Results, Kapoor and Picard [42].

The two HMMs are used to compute the probability of a sequence of N consecutive observations. More details are provided in their paper. They also established that 10 observations was sufficient to detect both slow and subtle head nods/shakes. The recognition rate for head nods was 81. 08%, while that for head shakes was 75.0%. The combined recognition rate was 78.46%.

3.4 Summary

In this chapter, a broad range of systems for FER have been presented. Focus was placed on research that took existing results and tweaked their processes or experimented in an instructive manner to improve the results. Attention was focused on the improvement of Gabor features, and on finding results that had the potential to improve the existing SASL Gabor-SVM FER system, or was able to at least equal its performance. While there were various combinations that appeared to improve recognition, they were unfortunately, isolated and inconclusive in suggesting a method better than that of the Gabor-SVM combination. Until such time when these improvements are corroborated independently, the SASL current system will suffice – particularly for the purposes of this thesis. There was no conclusive empirical data on noise in SVM classification, and the application thereof to the field of FER.

Chapter 4

Support Vector Machines

4.1 Introduction

Action units (see Chapter 2) must be classified as present or absent in images of the face. For this, a learning system is needed that can learn to determine the presence of an action unit. An image that might exhibit the presence of an action unit must be - by the learning system - placed into one of two classes in view of an image: present or absent. For this a *Support Vector Machine,* (SVM), [72; 45; 79] will be used.

Kernel methods use a set of functions to label data into classes. Support Vector Machines are kernel-based methods used for classification and regression. SVMs are particularly interesting because of their ability to change kernel methods depending on the nature of data being classified. SVMs exhibit flexibility through use of the *kernel trick* in which the default linear kernel is replaced with a *radial-basis* kernel, a *polynomial* kernel, *sigmoidal* kernel, and many other kernels. The training time of a SVM does not suffer from large feature vectors.

4.2 Premise

Suppose there is a set A of real space data vectors that exhibit a property binary in nature. A can then be divided into two categories A^+ and A^- depending on the presence and absence of that property. Suppose also that these sets are linearly separable. The intention in constructing a partition between the subsets A^+ and A^- is to be able to make deductions as to which category (*class*) uncategorized (*unlabeled*) data vectors belong to by determining which side of the hyperplane they lie on. By creating this separating partition, a *decision boundary* is created. The boundary that

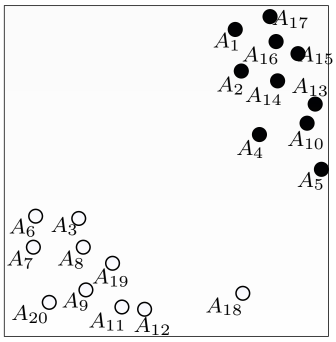


Figure 4.1: The data vectors in A^+ are clustered around the bottom, left corner, while those of A^- are clustered in the top, right corner.

best separates the two classes is referred to as the *maximal separating hyperplane*.

The process of locating this maximal separating hyperplane is known as *training* a support vector machine. A support vector machine is created when the maximal separating partition is found. When an unclassified data vector is presented to the support vector machine, it determines on which side of the maximal separating hyperplane the vector lies and labels the vector accordingly. The mathematics behind support vector machines will be described briefly in the following 3 sections.

4.3 Training

Suppose there is a set, A, of n data vectors in $\mathbb{R}^d$, where d is the dimensionality of the data vectors. Let each data vector in A be indexed by i, where i runs from 1 to n. It can thus be said $A = \{A_i | i \in \mathbb{N}, i \leq n\}$. Suppose also that every A_i falls into either of two classes $+1$ or -1, and that the classes are linearly separable so that a hyperplane H can be formed between them. Let the class that A_i falls into be denoted by y_i where $y_i \in \{-1, +1\}$. Each classified vector can be written as $\{A_1, y_1\}, \{A_2, y_2\}, ..., \{A_n, y_n\}$.

Let A^+ and A^- be defined as those vectors in A which fall into the $y = +1$ and $y = -1$ classes respectively: $A^+ = \{A_i | y_i = +1\}$ and $A^- = \{A_i | y_i = -1\}$. Each A_i being a d-dimensional data vector, can be written $A_i = (A_{i1}, A_{i2}, \ldots, A_{id})$. Suppose that the sets $A+$

Figure 4.1 shows a simplistic dataset in $\mathbb{R}^2$ where the black circles represent the A_i with $y_i = -1$ and hollow circles represent A_i with $y = +1$.

The following observation can be made: all A_i which lie either on H^+ or H^- are known as support vectors.

The remaining A_i lie either on H^+ or H^- or further away from H. H depends solely on the training vectors whose $\alpha_i > 0$. As long as these A_i remain closest to H, the SVM will remain the same (even if the others are moved

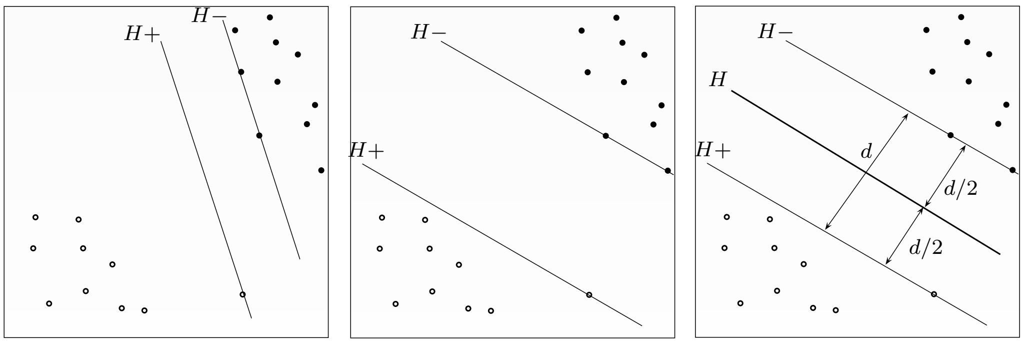


Figure 4.2: H^- and H^+ are tangential and parallel hyperplanes for A^- and A^+ respectively. The margin, d is the distance between H^- and H^+. The margin completely separates the sets A^- and A^+ in the left image, but only in the center and rightmost images is the margin maximized. The rightmost image shows the location of the maximal separating hyperplane H - halfway between and parallel to H^+ and H^-.

or removed).

The points in A^+ and A^- that lie closest to H form the hyperplanes H^+ and H^-. Infinitely many tangential and parallel hyperplanes H^+ and H^- can be constructed that pass through the support vectors in each set A^+ and A^- respectively. The maximal separating hyperplane is that which lies exactly midway between the tangential hyperplanes of each set. The SVM aims to locate the maximal separating hyperplane.

The distance from the maximal separating hyperplane to each tangential hyperplane H^+ and H^- is denoted by d^+ and d^- respectively. The distance between the tangential hyperplanes H^+ and H^- is denoted by d and is called the *margin*. $d = d^- + d^+ = 2d^+$. The maximal separating hyperplane is unique and maximizes the margin between H^+ and H^-. The larger the margin, the more clearly defined the difference between A^+ and A^- and thus the more accurately the SVM will be able to classify new data. This concept is best illustrated in Fig 4.2.

Various methods for constructing the hyperplane exist. They depend on the nature of the data provided for training and determine the type of SVM that must be used. The data provided for training lies in what is known as the *input space*. If the training set is linearly separable then the hyperplane can simply be constructed in the input space. This type of SVM is known as a *linear SVM*.

A hyperplane can be described by its general equation $\omega \cdot x + b = 0$. The vector $\omega \in \mathbb{R}^d$ is the normal and lies perpendicular to the plane while the parameter $b \in \mathbb{R}$ defines the offset. Since H^+ and H^- are parallel, ω will be the same for both hyperplanes. Altering b moves the plane along ω.

The tangential hyperplanes can be simplified to:

$$H^+ \ : \ \omega \cdot A_i + b = +1 \tag{4.1}$$

$$H^- \ : \ \omega \cdot A_i + b = -1 \tag{4.2}$$

and the maximal separating hyperplane is defined by:

$$H \ : \ \omega \cdot A_i + b = 0 \tag{4.3}$$

H^+ and H^- contain the support vectors of A^+ and A^- respectively, and all data points not on these two tangential hyperplanes must lie further from H. To calculate H, $\omega \in \mathbb{R}^d$ and $b \in \mathbb{R}$ need to be calculated such that

$$A_i \cdot \omega + b \geq +1 \quad \forall i|(y_i = +1) \tag{4.4}$$

$$A_i \cdot \omega + b \leq -1 \quad \forall i|(y_i = -1) \tag{4.5}$$

For every instance of A_i, ω and b must simultaneously satisfy the appropriate one of Eq. 4.4 and Eq 4.5. Combining Eq. 4.4 and Eq 4.5 yields the following inequality.

$$y_i(A_i \cdot \omega + b) \geq 1 \quad \forall A_i \in [1..n] \tag{4.6}$$

In addition to the constraint in Eq. 4.6, the margin, d, must be maximized between H^+ and H^-. Since H^+ and H^- are parallel, the distance d can be found as follows: The distance between hyperplanes can be found by taking vectors v^+ and v^-, parallel to ω (perpendicular to the plane) that passes through the origin. Subtracting v^- from v^+:

$$|v^+| = \frac{1-b}{|\omega|} \text{ and } |v^-| = \frac{-1-b}{|\omega|}\text{, so,} \tag{4.7}$$

$$d = |v^+| - |v^-| = \frac{1-b}{|\omega|} - \frac{-1-b}{|\omega|} = \frac{1-b+1+b}{|\omega|} \quad = \frac{2}{|\omega|} \tag{4.8}$$

It follows that, since H lies midway between H^+ and H^-:

$$d^+ = d^- = \frac{1}{|\omega|} \tag{4.9}$$

Maximising the margin d requires the maximisation of $\frac{2}{|\omega|}$. This in effect requires a minimisation of $|\omega|$, or equivalently, $\frac{1}{2}|\omega|^2$, subject to the constraints of Eq. 4.6 (the factor of $\frac{1}{2}$ is added for mathematical convenience). This is a constrained optimization problem. The Lagrangian method is commonly used for solving such problems. This method seeks to minimise an *objective function* under a set of constraints by creating a new function called the Lagrangian. The use of Lagrange multipliers serves to simplify this constrained optimisation problem. It works by assigning non-negative constants (Lagrange multipliers) to each constraint equation and scaling the constraint accordingly. Instead of finding the minima of the constrained $\mathbb{R}^d$ function $f = \frac{1}{2}|\omega|^2$ with l constraints (for each A_i), f is an unconstrained function in $\mathbb{R}^{d+l}$. The scaled constraints form the Lagrangian when subtracted from the objective function. The Lagrangian function can be written as:

$$L(\mathbf{x}, \alpha) = f(\mathbf{x}) - \sum_{i=1}^{n} \alpha c_i(\mathbf{x}) \tag{4.10}$$

where $f(\mathbf{x})$ is the objective function, α_i are the Lagrangian multipliers and c_i are the constraints. Since c_i are inequality constraints, each component of α must be equal to or exceeding 0. This gives:

$$L(\omega, \mathbf{b}, \alpha) = \frac{1}{2}|\omega|^2 - \sum_{i=1}^{l} \alpha_i [y_i(\omega \cdot A_i + b) - 1] \tag{4.11}$$

$$= \frac{1}{2}|\omega|^2 - \sum_{i=1}^{l} \alpha_i y_i(\omega \cdot A_i) - b\sum_{i=1}^{l} \alpha_i y_i + \sum_{i=1}^{l} \alpha_i \tag{4.12}$$

This Lagrangian must be minimized with respect to w and b and maximized with respect to $\alpha \geq 0$. The Wolfe dual form is sometimes used to simplify the problem of solving the Lagrangian. The Wolfe dual can be applied to convex programming problems, provided both the objective functions and the constraint functions are differentiable. Wolfe's dual form rephrases the Lagrange problem: where the objective function is minimized subject to all constraints c_i for all i, the problem now is one of maximizing the Lagrangian subject to one constraint - that L is minimised with respect to x.

The solution to the former problem – the *primal* – occurs in the same place in $\mathbb{R}^{d+i}$ as that of the Wolfe dual; namely, the saddle point. It should be noted here that L is a quadratic function (see Eq. 4.12). One method of locating the saddle point is by finding the derivative of $L(\omega, \mathbf{b}, \alpha)$ and solving it for all points where $\frac{\partial}{\partial\omega}L$ and $\frac{\partial}{\partial b}L$ equal 0.

$$\nabla\omega = \omega - \sum_{i=1}^{l}(\alpha_i y_i A_i) = 0 \tag{4.13}$$

$$\omega = \sum_{i=1}^{l}(\alpha_i y_i A_i). \tag{4.14}$$

Differentiating with respect to b yields Eq. 4.15:

$$\nabla\omega = \sum_{i=1}^{l}(\alpha_i y_i) = 0 \tag{4.15}$$

Substituting Eq. 4.14 and 4.15 into Eq. 4.12:

$$L(\omega, b, \alpha) = \frac{1}{2}\left(\sum_{i=1}^{l}\alpha_i y_i A_i\right)^2 - \sum_{i=1}^{l}\alpha_i y_i\left[\left(\sum_{j=1}^{l}\alpha_j y_j A_j\right)\cdot A_i\right] - b\,(0) + \sum_{i=1}^{l}\alpha_i \tag{4.16}$$

$$= \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j A_i \cdot A_j \tag{4.17}$$

Because w has dropped away, Eq. 4.16 can be rewritten as:

$$L_w(b, \alpha) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j A_i \cdot A_j \tag{4.18}$$

L_w is the Wolfe dual form of L, the Lagrangian. At this stage, L_w needs to be maximized for the remaining variables α and b. The Karush-Kuhn-Tucker (KKT) conditions have been shown to be sufficient in solving for b [9; 32]. The KKT conditions for a local minimizer apply when solving for local minima (see Eq. 4.15 and Eq. 4.14). The KKT conditions are:

$$\frac{\partial}{\partial\omega_j}L_\omega = \omega_v - \sum_{i=1}^{l}\alpha_i y_i A_{iv} \quad \text{for } \ j = 1, \ldots, d. \tag{4.19}$$

$$\frac{\partial}{\partial b}L = -\sum_{i=1}^{l}\alpha_i y_i = 0 \tag{4.20}$$

$$y_i(A_i \cdot \omega + b) - 1 \geq 0 \quad \text{for } \ i = 1, \ldots, l \tag{4.21}$$

$$\alpha_i \geq 0 \quad \forall i \tag{4.22}$$

$$\alpha_i(y_i(L \cdot A_i + b) - 1) = 0 \quad \forall i \tag{4.23}$$

Eq. 4.23 is referred to as the complementarity condition. Solving it using those A_i whose $\alpha_i > 0$ will yield b^+ and

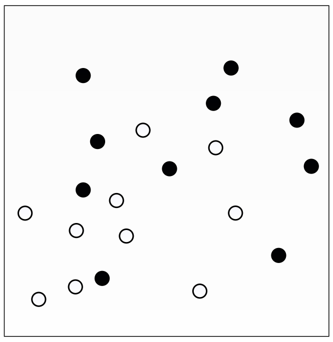


Figure 4.3: The data here is clearly not linearly separable.

b^-, which correspond to H^+ and H^- respectively.

$$\alpha_i(y_i(\omega \cdot A_i + b) - 1) = 0 \quad \text{then,} \tag{4.24}$$

$$y_i(\omega \cdot A_i + b) - 1 = 0 \tag{4.25}$$

$$b = \frac{1}{y_i} - (\omega \cdot A_i) \tag{4.26}$$

into which any training instance (A_i, y_i) can be substituted.

In practice b is averaged over all training vectors A_i. Thus

$$b = \sum_{i=1}^{l} \frac{\frac{1}{y_i} - (\omega \cdot A_i)}{l} \tag{4.27}$$

At this stage hyperplane H has both ω and b, and thus its definition.

4.4 Non-Separable Data

Not all data sets are linearly separable. Figure 4.3 shows a larger data set also in $\mathbb{R}^2$ where the training examples are not linearly separable. No hyperplane, or in the case of the set of data in Figure 4.3 ($\mathbb{R}^2$), no line, can completely separate the two classes. One of two methods can be used to solve this problem: the first involves introducing a margin of allowable error. The second involves describing the boundary using a more complex hyperplane. The former method is briefly described here.

Cortes and Vapnik [17] revealed a method for solving problems where the data is linearly inseparable in which a margin of error is introduced. Training data is allowed to violate the equalities $\omega \cdot A_i + b \geq +1$ and $\omega \cdot A_i + b \leq -1$. However, this violation must be allowed only when absolutely necessary, and when it does occur, a penalty must be imposed on the objective function, $\frac{1}{2}|\omega|^2$. For each A_i, a non-negative slack variable ξ_i is introduced. The definitions of the tangential hyperplanes change accordingly:

$$H^+ : \qquad \omega \cdot A_i \geq +1 - \xi_i \quad \forall i | y_i = +1 \tag{4.28}$$

$$H^- : \qquad \omega \cdot A_i \leq -1 + \xi_i \quad \forall i | y_i = -1 \tag{4.29}$$

If, at most, $n-2$ training examples are classified incorrectly, $\sum_{i=1}^{l} \xi_i$, the sum of all errors, can be used as the upper bound on the number of errors, and thus serves sufficiently as a penalty on the objective function when added to it. In practice, this penalty is scaled by factor C, according to the accuracy required in a particular setting. The higher C, the larger the penalty imposed when errors occur. Combining Eq. 4.28 and Eq. 4.29 yields:

$$y_i(\omega \cdot A_i + b) \geq 1 - \xi_i \quad \xi_i \geq 0 \tag{4.30}$$

and the new objective function[1]:

$$\frac{1}{2}|\omega|^2 + C \sum_{i=1}^{l} \xi_i \tag{4.31}$$

After introducing a second vector of Lagrange multipliers to ensure the non-negativitiy of ξ_i, the new Lagrangian function becomes:

$$L(\omega, b, \alpha, \xi, \mu) = \left(\frac{1}{2}|\omega|^2 + C \sum_{i=1}^{l} \xi_i \right) - (y_i(\omega \cdot A_i + b) - 1 + \xi_i) - \sum_{i=1}^{l} \mu_i \xi_i. \tag{4.32}$$

A similar procedure is followed for minimizing Eq. 4.32 as was used when the data set was linearly separable. Differentiating $L(\omega, b, \alpha, \xi, \mu)$ with respect to $\mathbf{w}$, b and now ξ shows:

$$\omega = \sum_{i=1}^{l} \alpha_i y_i A_i \tag{4.33}$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{4.34}$$

$$C - \alpha_i - \mu_i = 0 \tag{4.35}$$

[1]A more general objective function is used in practice. $\frac{1}{2}|\omega|^2 + C\left(\sum_{i=1}^{l} \xi_i\right)^k$, where $k = 1$ or 2. For pattern recognition problem, k is set to 1. [76]

Substituting Eq. 4.33, Eq. 4.34 and Eq. 4.35 into Eq. 4.32, L_w is simplified to:

$$L(\omega, b, \xi, \alpha, \mu) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j A_i \cdot A_j \tag{4.36}$$

The procedure for calculating b is exactly the same. b can be found by taking any point for which $0 < \alpha_i < C$ and by inserting it into the complementarity conditions $\alpha_i\{y_i(A_i \cdot \omega + b) - 1 + \xi_i\} = 0$.

4.5 Non-Linear Support Vector Machines

Some data sets are not linearly separable in the input space. In most cases the data become linearly separable when mapped into a higher dimensional space, called the feature space. The function that performs this mapping is known as a *kernel* function, and takes the form

$$K(A, \mathbf{Y}) = \Phi(A)\Phi(\mathbf{Y}) \tag{4.37}$$

Throughout this chapter, it can be observed that data vectors appear only in the form of inner products $A_i \cdot A_j$. As long as the result of the kernel transformation is an inner product, L_w will still be valid. For transformation $\phi : \mathbb{R}^d \longrightarrow H$, K will need to be substituted into all equations requiring the result of the inner product. The training and classification algorithms would only depend on the data through dot products in H. In its Wolfe-Dual form, the Lagrangian can be rewritten as:

$$L_w(b, \alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(A_i, A_j) \tag{4.38}$$

And the equation for classification is rewritten as:

$$y_i = \text{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i K(A_i, A_j) + b \right) \tag{4.39}$$

Various kernel functions exist. The mappings which can be applied are described by Mercer's condition [9]. Mercer's theorem states that if and only if, there exists any function $g(x)$ such that $\int(g(x)^2 dx)$ is finite, then,

$$\int K(x, y) g(x) g(y) dx dy \geq 0. \tag{4.40}$$

Three non-linear SVM kernels have been found to be of most use in practice [79]: the Polynomial kernel (Eq. 4.41),

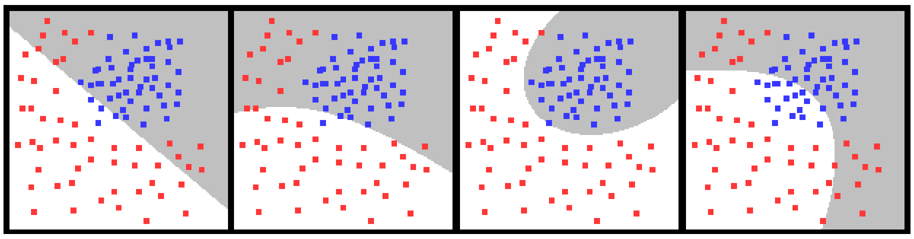


Figure 4.4: Kernel Types: From left to right, Linear, Polynomial, Radial Basis Function and Sigmoid. The maximally separating hyperplane lies on the border between the white and grey regions.

the Gaussian Radial Basis Function (RBF) kernel (Eq. 4.42) and the Sigmoid kernel[2] (Eq 4.43.)

$$K(x, y) = (x \cdot y + 1)^p \tag{4.41}$$

$$K(x, y) = e^{-\frac{|x-y|^2}{2\delta^2}} \quad \forall p \in N^+, p > 0 \tag{4.42}$$

$$K(x, y) = \tanh(kx \cdot y - \delta) \tag{4.43}$$

Kernels form a vector space and can be created as a result of performing various operations on the kernel. If K_1 and K_2 are kernels, then the following are kernels too:

- $K_1 + K_2$
- aK_1 for $a > 0$
- any linear combination of the above two operations.

There is currently no method of determining which kernel generalizes best for a particular problem. Figure 4.4 shows two sets of non-linearly separable data, red and blue, and the hyperplane as placed by different kernels. For this case of data, it is clear that the RBF kernel creates the best SVM.

4.6 Summary

This chapter briefly described the mathematical process of training and testing an SVM. Separable and non-separable data and the differences in hyperplane definition were discussed. Non-linear support vector machines were described, as well as the kernel trick, in which kernels can be interchanged as long as they meet Mercer's condition. The chapter ended with a brief look at the Polynomial kernel, the Gaussian RBF kernel and the Sigmoid kernel.

[2]The sigmoid kernel only satisfied Mercer's Condition for certain values of K and δ.

Chapter 5

Mathematical Methods

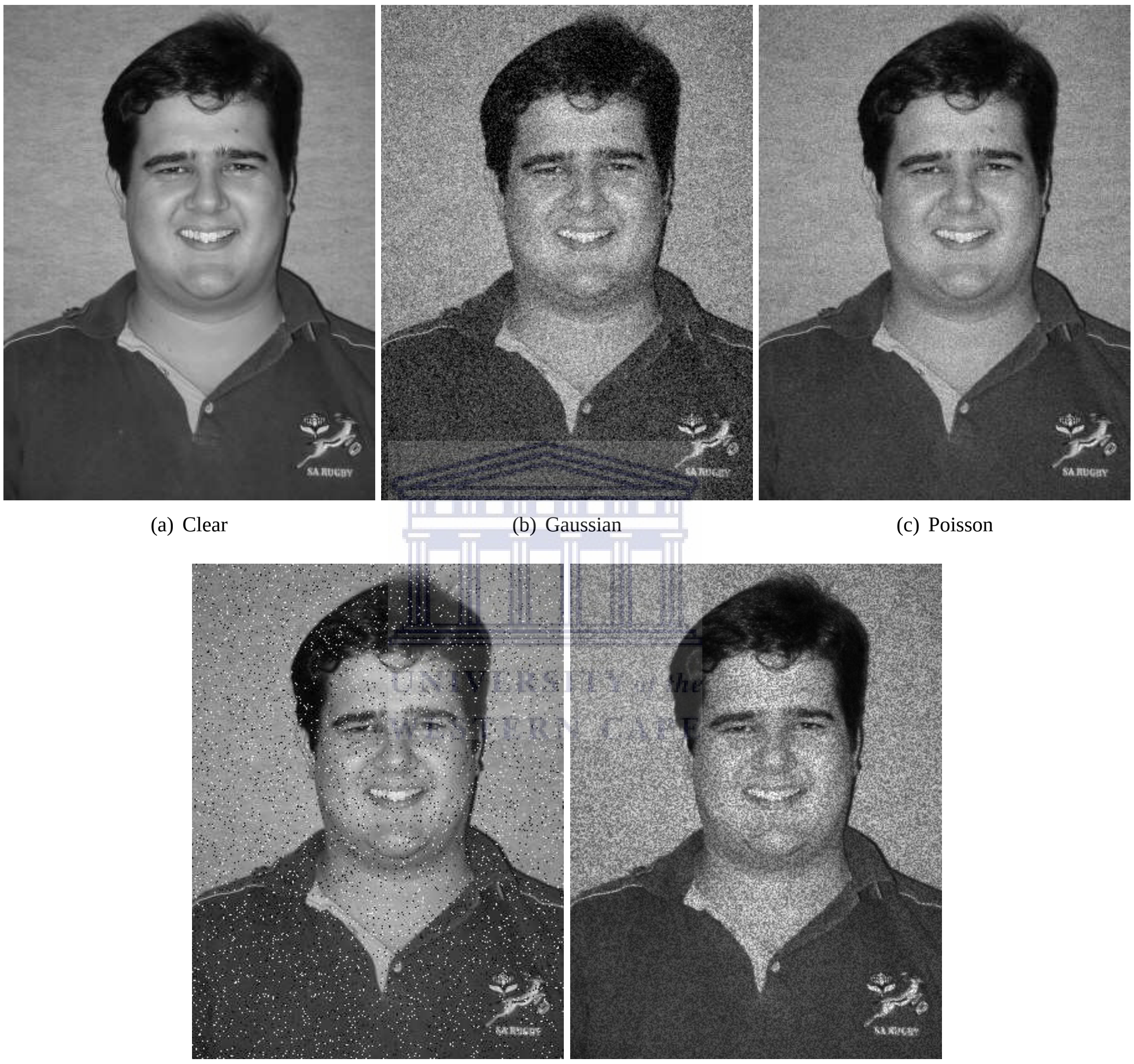
This chapter looks at the mathematical methods used in this thesis. This preprocessing includes normalization, noise addition and Gabor filtering. Before concluding, the metrics used in evaluating the experiments are also looked at.

5.1 Noise

Noise means any unwanted distortion of a wanted signal. Noise can block, distort, change or interfere with communication between both humans and electronic devices. Imaging sensors are affected by several factors, ranging from environmental conditions during image acquisition to transmission and storage to quality of sensing elements. For instance, when capturing images with a CCD camera, the amount of noise in images is greatly affected by light levels and sensor temperature. Images transmitted over a wireless network might be altered as a result of lightning or other atmospheric disturbance [35].

5.1.1 Noise Models

The prevalence of noise in signals and the need to reduce it, if not remove it completely, have led to modelling of different types of signal noise. There are several models of noise. To discuss them all would be beyond the scope of this thesis so only four common models are selected and analysed for their contribution to the classification accuracy of Gabor-preprocessed support vector machines for the task of facial expression recognition. The models selected here form the basis of most other noise models. They are Gaussian noise, Poisson noise, Speckle and Salt-&-Pepper noise. In Eqs. 5.1, 5.2, 5.3 and 5.4, x refers to the gray-level, a and b are fixed gray-level constants.


(a) Clear (b) Gaussian (c) Poisson

(d) Salt & Pepper (e) Speckle

Figure 5.1: A visual comparison of the different noise distributions: this figure shows an image in its original state (a), followed by the results after applying different noise distributions: (b) Gaussian, (c) Poisson, (d) Speckle, (e) Salt & Pepper

5.1.2 Gaussian Noise

Gaussian noise (Eq. 5.1) is the name given to noise which follows a Gaussian normal distribution function. The values the noise can take on are Gaussian-distributed. Gaussian noise is most commonly used as additive white noise to produce additive white Gaussian noise. The effects of applying Gaussian noise to an image can be seen in Fig 5.1.b. In Eq. 5.1, μ is the mean value and σ is the standard deviation.

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \mu \text{ = mean value, } \sigma \text{ = standard deviation} \tag{5.1}$$

5.1.3 Poisson Noise

The Poisson probability distribution function is given by Eq. 5.2. The parameter μ is not only the mean gray-value $\langle x \rangle$, but also its variance $\sigma_x^2 = \langle x^2 \rangle - \langle x \rangle^2$. These fluctuations in gray-value are called Poisson noise or (particularly in electronics) as shot noise. The effects of applying Poisson noise to an image can be seen in Fig 5.1.c.

$$p(x) = \frac{\mu^x e^{-\mu}}{x!} \quad \mu \text{ = expectation value} \tag{5.2}$$

5.1.4 Salt and Pepper Noise

Salt and pepper noise is also known as bipolar impulse noise. It appears in images as randomly occurring white and black pixels. Salt and pepper noise appears in images in situations where quick transients, such as faulty switching, occur. The effects of applying Salt And Pepper noise to an image can be seen in Fig 5.1.d. The probability distribution for Salt and Pepper noise is seen in Eq 5.3. For our experiments, $a = 0(black)$ and $b = 255(white)$

$$p(x) = \begin{cases} P_a & \text{for } x = a \\ P_b & \text{for } x = b \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

5.1.5 Speckle Noise

Speckle noise occurs in coherent imaging such as ultrasound imaging. It is formed with coherent radiation of a medium that contains sub-resolution scatterers. Eq. 5.4 shows the Speckle Noise probability distribution. It increases the pixel value of a local area within the image. It exists in active radar and Synthetic Aperture Radar (SAR) images. The effects of applying Speckle noise to an image can be seen in Fig 5.1.e.

$$p(x) = \frac{e^{\frac{-x^2}{\sigma^2}}}{\sigma^2} \tag{5.4}$$

5.2 Gabor Filters

Gabor filters are filters that, when applied to waveforms, exhibit localization properties in both space and frequency domains. That is, they serve to isolate and amplify components of a waveform that lie close to a pre-selected frequency. Gabor filters have two components. They comprise of a complex sinusoidal function, referred to as the *carrier*, which is modulated by a Gaussian function, referred to as it's *envelope*. Images are 2-D waveforms, so only the 2-D form of a Gabor filter is presented here [30]:

$$\mathbf{g}(x, y) = \mathbf{s}(x, y)\mathbf{w_r}(x, y) \tag{5.5}$$

where $\mathbf{s}(x, y)$ is the complex sinusoidal and $\mathbf{w_r}(x, y)$ is the Gaussian envelope.

5.2.1 Complex sinusoidal

The carrier can be defined as:

$$\mathbf{s}(x, y) = e^{j(2\pi(u_0 x + v_0 y) + P)} \tag{5.6}$$

where u_0 and v_0 is the *Gabor filter spatial central frequency* and P defines the phase of the sinusoidal. $\mathbf{s}(x, y)$ can also be expressed as two separate real functions. Recall that $e^{j\theta} = \cos(\theta) + j\sin(\theta) = S_{real} + jS_{imag}$ where

$$S_{real} = \cos(2\pi(u_0 x + v_0 y) + P) \tag{5.7}$$

$$S_{imag} = \sin(2\pi(u_0 x + v_0 y) + P) \tag{5.8}$$

where u_0, v_0 defines the spatial central frequency of $\mathbf{s}$ in cartesian coordinates.

5.2.2 Gaussian Envelope

The Gaussian envelope specifies the spatial locality of the waveform to be filtered. It can be defined as:

$$\mathbf{w_r}(x,y) = ke^{-\pi(a^2(x-x_0)_r^2)+b^2(y-y_0)_r^2} \tag{5.9}$$

The peak lies at (x_0, y_0) and a and b are scaling parameters, and r stands for the following rotation operations:

$$(x-x_0)_r = (x-x_0)\cos(\theta) + (y-y_0)\sin(\theta) \tag{5.10}$$

$$(y-y_0)_r = -(x-x_0)\sin(\theta) + (y-y_0)\cos(\theta) \tag{5.11}$$

5.2.3 Gabor Filter Function

Combining Eq. 5.10 and Eq. 5.11, the Gabor function can be rewritten as:

$$\mathbf{g}(x,y) = ke^{-\pi(a^2(x-x_0)_r^2+b^2(y-y_0)_r^2)}e^{j2\pi(u_0x+v_0y)+P} \tag{5.12}$$

The Gabor function is complex and has the following parameters:

- K: scaling magnitude of the envelope
- a, b: scaling magnitude of the X and Y axes of the envelope
- θ: rotation angle of the envelope
- (x_0, y_0): location of the envelop's peak
- (u_0, v_0): spatial frequencies of carrier
- P: Phase of carrier

Upon translation of **g** to the frequency domain where it becomes $\mathbf{G}$, the following is noted:

$$\mathbf{G}(u,v) = \frac{k}{ab}e^{-j(-2\pi(x_0(u-u_0)+y_0(v-v_0))+P)}.e^{-\pi(\frac{(u-u_0)_r^2}{a^2}+\frac{(v-v_0)_r^2}{b_2})} \tag{5.13}$$

Also to be noted is Eq. 5.14:

$$\mathbf{G}(u,v) = \mathbf{W}(u-u_0, v-v_0) \tag{5.14}$$

That is, the frequency response of $\mathbf{g}(x,y)$ (Gabor filter) is simply the frequency response of $\mathbf{g}(x,y)$ (the Gaussian envelope) shifted to point (u_0, v_0).

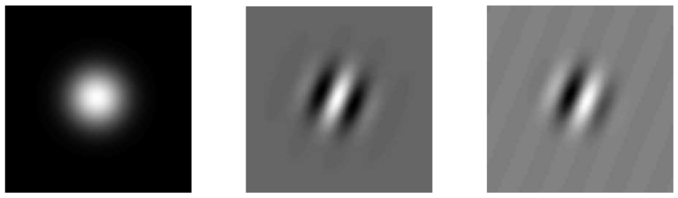


Figure 5.2: The absolute value, The real and imaginary parts of a complex Gabor filter. The images are 64X64 pixels.

5.2.4 Polar Coordinates

The equivalent of Eq. 5.12 in polar coordinates for $(u_0, v_0) \rightarrow (F_0, \omega_0)$ is given by Eq. 5.15, and its magnitude and phase given by Eq. 5.16 and Eq. 5.17 respectively:

$$g(x,y) = e^{j(2\pi F_0(x\cos(\omega_0)+y\sin(\omega_0))+P)} \tag{5.15}$$

$$\text{magnitude}(G(u,v)) = \frac{k}{ab}.e^{-\pi(\frac{(u-u_0)_r^2}{a^2}+\frac{(v-v_0)_r^2}{b^2})} \tag{5.16}$$

$$\text{phase}(G(u,v)) = -2\pi(x_0(u-u_0)+y_0(v-v_0))+P) \tag{5.17}$$

5.2.5 Application

Application of Gabor filters most commonly takes place in the frequency domain due to ease of computation. The Fourier transform of the input waveform is multiplied by the Fourier transform of the filter. This process results in the *Gabor decomposition* of the waveform. After multiplication, the product is inverse transformed into the space domain. Altering the spatial frequency and orientation tunes the filter into particular frequencies. This affords it the ability to select any of the full range of frequencies in a waveform.

Often in FER, a set, referred to in literature as a *bank*, of Gabor filters are applied to an image. Each filter is tuned to a different frequency and orientation. The filtered images are then catenated to form a Gabor jet. This jet has often proven to be richer in information than single decompositions alone. Several papers have been released suggesting the most appropriate filter banks for various purposes. However there is no conclusive evidence to say that any one configuration is better than another. Whitehill [79] suggests that optimal filter bank size depends on the specific application. For facial expression recognition the norm is to use a bank of 40 filters, each one differing in peak frequency by $\frac{\pi}{8}$ radians as well as 8 orientations [79; 30; 20].

5.2.6 Uses of Gabor filters

Gabor filters have been applied successfully in numerous learning applications. These include face [7] and facial expression [79] recognition, texture segmentation [74], script recognition [68], fingerprint recognition[75], iris recognition[82], and many more. [48] used Gabor filters to extract features from CT images for the purpose of recognizing liver diseases. Gabor filters have been shown to emulate orientation selectivity that is a characteristic common to an area of the brain known as the striate cortex[71]. This characteristic has been most commonly noticed in cats as well as a species of monkey known as the macaque. In macaque monkeys, this ability has been isolated further to the layer VI cells in the striate cortex. Gabor filters enhance skin bulges and other appearance changes induced by facial expressions [79]. Gabor filters have also been shown to emulate simple cortical cells in humans [59].

5.3 Metrics

Some action units occur so rarely that in a dataset where none of a particular action unit is present (0% hit rate), the percentage of images classified correctly would still be high if the classifier simply returned false for all images. Thus, for the task of FER, accuracy representation by percentage classified correctly can be misleading. A more robust metric for recognition accuracy is the A' statistic. This refers to the area under a Receiver Operator Characteristics (ROC) curve.

ROC curves were developed as a by-product of research into signal detection theory when engineers in World War 2 deciphered noise-contaminated radio signals. *Sensitivity* measures the proportion of correctly identified positives, while *Specificity* measures the proportion of correctly identified negatives. ROC curves are graphical plots of the sensitivity versus (1- specificity) for a binary classifier system as its discrimination threshold is varied. In FER, sensitivity of one SVM trained to classify a particular action unit would be the percentage of peak facial images containing that action unit. The percentage of neutral facial images identified as not having the action unit would determine the specificity for that SVM. A sensitivity of 100% means that the classifier correctly identifies all actual positives. A sensitivity test that yields negative results will rule out the presence of the action unit.

The results of one of the binary classifiers are:

- True positive: Peak images correctly classified as having an action unit present.
- False positive: Neutral images incorrectly classified as having an action unit present.
- True negative: Neutral images correctly classfied as not having the action unit present.
- False negative: Peak images incorrectly classified as having the action unit present.

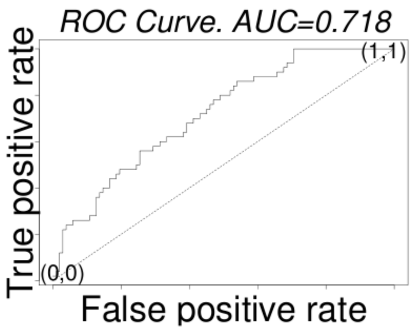


Figure 5.3: ROC Space. The dotted line indicates the ROC curve which, if matches the diagonal, means the performance of the classifier is as good as a random guess [16].

$$\textit{specificity} = \frac{\text{number of True Negatives}}{\text{number of True Negatives} + \text{number of False Positives}} \tag{5.18}$$

$$\textit{sensitivity} = \frac{\text{number of True Positives}}{\text{number of True Positives} + \text{number of False Negatives}} \tag{5.19}$$

A specificity of 100% means that the test recognizes all actual negatives - for example, all peak images will be recognized as having the action unit present. From Eq 5.18 it is clear that 100% specificity can be achieved by a test that determines that all images have a particular action unit whether that action unit is present or not. Thus, the specificity alone does not reveal how well the test recognizes positive cases. The sensitivity of the test is also needed.

The Area Under Curve (AUC) is the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one[31]. It can be shown that the area under the ROC curve is closely related to the MannWhitney U statistic, which tests whether positives are ranked higher than negatives. It is also equivalent to the Wilcoxon test of ranks. The A' metric incorporates both true positive and false positive rates of a classifier and is the metric of choice for presenting results.

5.4 Summary

This chapter looked at the preprocessing components of the facial expression recognition system that is used in this thesis. The preprocessing that is involved in firstly adding various types of noise was described. This was followed by a brief look at the makings of a Gabor filter. ROC curves and evaluation metrics were looked at before concluding.

Chapter 6

Experimental Results

This chapter introduces the preprocessing that takes place before images are trained and classified. The experimental procedure is then covered, in which four broad experiments are described. The results of these experiments are then presented and discussed.

6.1 Preprocessing

Selection of Images

The system was tested on a database set up by CMU[40]. The database contains image sequences of facial expressions performed by 97 subjects. Each subject performs at least 2 expressions starting from the neutral position and peaking with the expression. The setup of photos are such that there is one video camera directly facing the subject, and subjects make all expressions looking directly into the camera. In-plane rotation is less than 5 degrees and out-of-plane rotation is negligible. Each image was presented to a human FACS coding expert who studied them and logged the AUs present in that image. Images were selected for use according to the AUs present; AUs were only selected for experimentation if there were at least 50 images in which they were present. The AUs for which this was the case were: 1, 2, 4, 5, 6, 7, 9, 11, 12, 15, 17, 20, 23, 24, 25, 26, and 27. Two neutral and two peak images were used for each expression for which an SVM was trained.

AU Statistics

Table 6.1 shows the number of images in the database that each AU had for use in the system. The more images are available for training, the better the SVM should generalize over the test sets. AU 9, AU 17 and AU 25 had more than

Table 6.1: Number of Images in the Database for each AU

Number of images	Action Units
50-99	9,11,23,24,26
100-150	15,20
150-199	2,5,27
200-299	1,6,7,12
300+	4,17,25

300 images, and thus their results can be regarded as the most general.

Noise Addition

The Matlab [60] Image Processing Toolbox is used to add noise to images when required. Depending on the experiment being performed noise is added to the images. 15 versions of the original database, each with a different noise model applied to it, were created before training the SVMs.

Normalization

The images were normalized: they are cropped and scaled proportionally into squares of 64 by 64 pixels, so that the images are centered around the face, the inter-occular distance is 24 pixels, and vertical eye-mouth distance is 26 pixels. To simplify the process, all images are normalized before training SVMs begins.

Gabor Filtering

The process of filtering is computationally simpler in the frequency domain. Images are Fourier transformed into the frequency domain where they are multiplied by the Fourier transformed Gabor Filter. The filtered images are then inverse transformed back to the space domain. Matlab's Image Processing Toolkit is used for Fourier transformation.

A bank of 40 Gabor filters is used, at five spatial frequencies and eight orientations. The 40 filters applied to each image produce 40 filtered images, which are then concatenated together to form a 64 X 64 X 40 element long *Gabor jet*. Feature vectors are calculated as the complex magnitudes of the values in the Gabor jets. The vectors were subsampled by a factor of 16 and normalized to unit length.

Cross-validation

When supplied with a large dataset, 10-fold cross-validation is often used as a means to ensure that all data instances are used for both training a learning engine as well as testing it. The data is split into 10 folds or sets. The learning

```
-1 1:1 6:1 17:1 19:1 39:1 42:1 53:1 64:1 67:1 73:1 74:1 76:1 80:1 83:1
-1 2:1 6:1 18:1 20:1 37:1 42:1 48:1 64:1 71:1 73:1 74:1 76:1 81:1 83:1
+1 5:1 11:1 15:1 32:1 39:1 40:1 52:1 63:1 67:1 73:1 74:1 76:1 78:1 83:1
```

Figure 6.1: 3 sample training instances. Each line is one data instance. The first number is the class in which each data instance falls.

engine is then trained 10 times, each time changing the fold that it is tested on and using the remaining 9 sets as training data. This process ensures that every fold is both trained and tested on. The average of the 10 experiments then becomes the final result of training that learning engine. 10 fold cross-validation was performed during all experiments.

Support Vector Machines

LibSVM [10] is a popular support vector machine library. LibSVM receives files formatted as shown in Figure 6.1. In this case, lines starting with“-1” indicates that the Gabor jet of an image in which the AU is not present follows, while lines starting with “+1” indicates that the Gabor jet of an image in which the AU is present follows. Every line in the training and test files begin with either the “+1” or“-” and then has up to 10720[1] numbers following it. LibSVM requires training and test instances to be separated into different files.

ROC Curve Generation

A popular python libSVM script, known as *plotroc.py* [10], is used to create ROC curves. Plotroc is compatible with LibSVM. Plotroc creates a support vector machine with the training information and tests it on the test datafile provided. Plotroc can be run in two ways. The first uses only a training datafile and performs *n*-fold cross validation on it. The second requires it to be run using both training and test datafiles. In the experiments performed in this thesis, both training and test data files were supplied, and the data contained in each fold manually controlled. The statistic used from the ROC curves is the AUC value.

6.2 Experimental procedure

Five experiments are described in this section. The first is a control experiment, which sets the standard of classification for the following experiments. This experiment creates support vector machines that are both trained and tested on clear images. The second, third and fourth experiments introduce noise into the equation. Experiment 2 takes the same

[1](64 X 64 X 40)/16 where 16 comes from the subsampling process

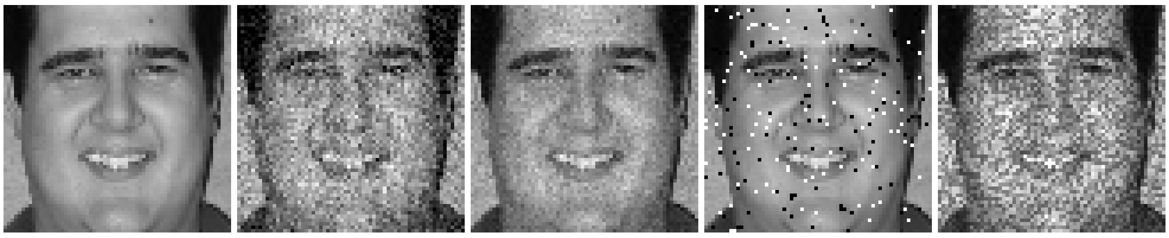
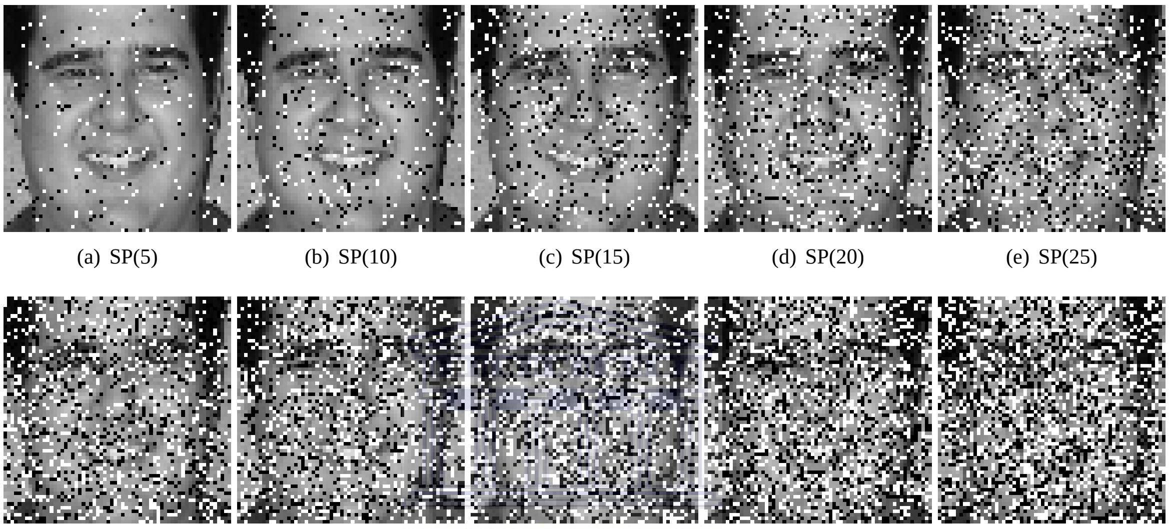

(a) Clear (b) Gaussian (c) Poisson (d) Salt & Pepper (e) Speckle

Figure 6.2: Original, Gaussian, Poisson, speckle, Salt & Pepper Noise Filtered Images


(a) SP(5) (b) SP(10) (c) SP(15) (d) SP(20) (e) SP(25)

(f) SP(30) (g) SP(35) (h) SP(40) (i) SP(45) (j) SP(50)

Figure 6.3: Degrees of salt And pepper noise, from 5% to 50%

support vector machines that were created in Experiment 1, and tests them on noisy versions of the test sets used for Experiment 1. Experiment 3 trains classifiers with noisy images, and tests them on clear images, while Experiment 4 trains and tests the classifiers on noisy images only. All noise filters are cross-tested. That is, for the experiments that include noise, all different types of noise are used in training SVMs, which are then tested on all other types of noisy images.

Figure 6.2 shows the broad category noise groups of normalized images. From right to left, the image filters are: clear/none, Gaussian, Poisson, salt and pepper and speckle. Figure 6.3 shows the range of salt and pepper noise normalized images.

6.2.1 Experiment 1: Setting up a Classification System

The aim of this experiment was to create a Gabor-SVM based FACS classifier. Using the exact procedure described in Section 6.1, a set of support vector machines were created, each one trained to determine if a particular action unit is present in an image. These support vector machines are trained and evaluated on images not containing any noise. This experiment forms the basis for the following experiments and serves to determine a benchmark for classification accuracy. A brief summary of the process follows:

* for each action unit (17)
    * train a SVM using clear images
    * test the SVM using clear images
    * perform 10-fold cross validation (10), average the 10 AUC values generated
* Total Experiments: 170.

6.2.2 Experiment 2: Training Without Noise, Classifying With Noise

This experiment assumes clean, normalized images are used for the training of the classifiers. These classifiers are then tested on images in which varying levels of noise are present. 13 types of noise are used: Gaussian, Poisson, speckle, salt & pepper 5, salt & pepper 10, salt & pepper 15, salt & pepper 20, salt & pepper 25, salt & pepper 30, salt & pepper 35, salt & pepper 40, salt & pepper 45, salt & pepper 50, A brief summary of the process follows:

* for each action unit (17)
    * train a SVM using clear images
        * for each noise type (13)
            * test the SVM using images with that noise
            * perform 10-fold cross validation (10), average the 10 AUC values generated.
* Total experiments: 2210.

6.2.3 Experiment 3: Training With Noise, Classifying Without Noise

The purpose of this experiment is to determine whether support vector machines that are trained on noisy data can cope with clear images. Images with noise have less information than those without. The result of this experiment will show if training an SVM with less information produces good results when tested on images that have more information. A brief summary of the process follows:

* for each action unit (17)
    * for each noise type (13)
        * train a SVM using images with that noise

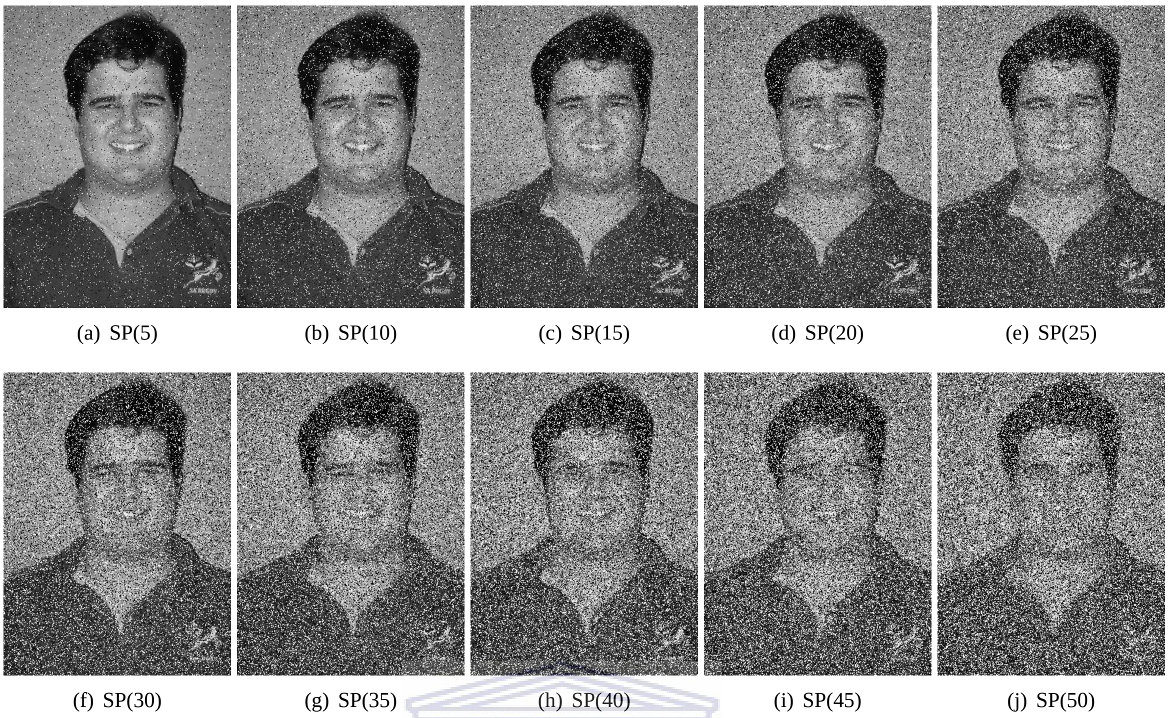


Figure 6.4: Increasing the amount of Salt And Pepper noise in 5% intervals.

- test the SVM using clear images
- perform 10-fold cross validation (10), average the 10 AUC values generated

* Total experiments: 2210.

6.2.4 Experiment 4: Training With Noise, Classifying With Noise

This final experiment trains SVMs using images with all types of noise, and then tests them using images with all types of noise. A brief summary of the process follows:

* for each action unit (17)
    * for each noise type (13)
        * train a SVM using images with one type of noise
        * test the SVM on images that have every other type of noise (12)
        * perform 10-fold cross validation (10), average the 10 AUC values generated
* Total experiments: 26520.

Table 6.2: Interpreting AUC values

AUC	Conclusion
$= 0.5$	No discrimination
$0.7 \leq AUC < 0.8$	Acceptable discrimination
$0.8 \leq AUC < 0.9$	Excellent discrimination
$AUC \leq 0.9$	Outstanding discrimination

6.2.5 The Acceptance-Level Experiment

As well as evaluating classification performances of the SVMs on noisy images, a noise acceptance-level experiment is performed, which serves to experimentally decide the threshold at which noise levels completely impede performance. In this experiment the amount of Salt and Pepper noise is gradually increased until 50% of the image is obscured. Each SVM is trained using images with one type of noise, and tested on each set of Salt & Pepper images and in this way, it is possible to determine at which levels the classification performance becomes unacceptable. Figure 6.4 shows how an image changes as more Salt and Pepper noise is added.

6.3 Results

This section presents the results obtained during each train-test phase. Results are in the form of AUC values, obtained from the ROC curves generated during every stage of testing support vector machines. General guidelines for interpreting AUC values are listed in Table 6.2. An AUC equal to 0.5 means that the probability of the SVM model correctly identifying the action unit is equivalent to flipping a coin. In order to save space, full result tables are presented in the Appendix A. Table 6.4 shows the average AUC values over all action units. Table A.1 shows the medians for the AUC results and Table A.2 shows the sample standard deviations.

For ease of writing, the terms *clear-*, *Gaussian-*, *Poisson-*, *S & P-* and *speckle-* images are used to refer to an that have had no, Gaussian, Poisson, salt & pepper or speckle noise added to them respectively. Furthermore, the terms *clear-SVM*, *Gaussian-SVM*, *Poisson-SVM*, *S & P-SVM* and *speckle-SVM* are used to refer to support vector machines that have been trained using Gabor-filtered clear-, Gaussian-, Poisson-, salt and pepper-, and speckle- images respectively.

6.3.1 Experiment 1: Train on Clear, Test on Clear

Table 6.3 shows the average AUC values over 10 folds for each action unit. Figure 6.5 shows the results in a graphical form. The mean AUC value over all action units is 0.957. The median is 0.988, while the standard deviation is 0.084.

Table 6.3: Average AUC for each Action Unit

AU #	**AUC**
Brow AUs	
1	0.999
2	0.997
4	0.999
Eye AUs	
5	1.000
6	1.000
7	0.979
Nose AUs	
9	1.000
Mouth AUs	
11	1.000
12	0.994
15	0.957
17	0.946
20	0.988
23	0.870
24	0.914
25	0.967
26	0.661
27	0.996
mean	**0.957**
median	**0.988**
stdev	**0.084**

A high spread of AUC values reflects inconsistencies in the classification performance and possibly in the data. In this experiment, the spread is extremely low, indicating high consistency between results.

Figure 6.5 shows consistently good results for action units around the brows, eyes and nose. These are AU 1, AU 2, AU 4, AU 5, AU 6, AU 7 (which has the lowest AUC in the upper face) and AU 9. Action units around the mouth, however, have a wider range of AUC values. The lowest performing action unit is AU 26 which, from Table 2.2, is the jaw drop. One possible explanation for the poor performance of the mouth AUs is that the mouth region exhibit greater variability in bulges and wrinkles of the skin [79].

6.3.2 Experiment 2: Train on Clear, Test on Noisy

Figure 6.6 shows a graph of AUC values when SVMs are trained using clear images and tested on noisy images. On average over all AUs, the results for testing Clear images against Poisson images were 0.993 - the highest of all, and even higher than Clear against Clear. This is an unexpected outcome. Gaussian and speckle images also performed better than Clear images in testing.

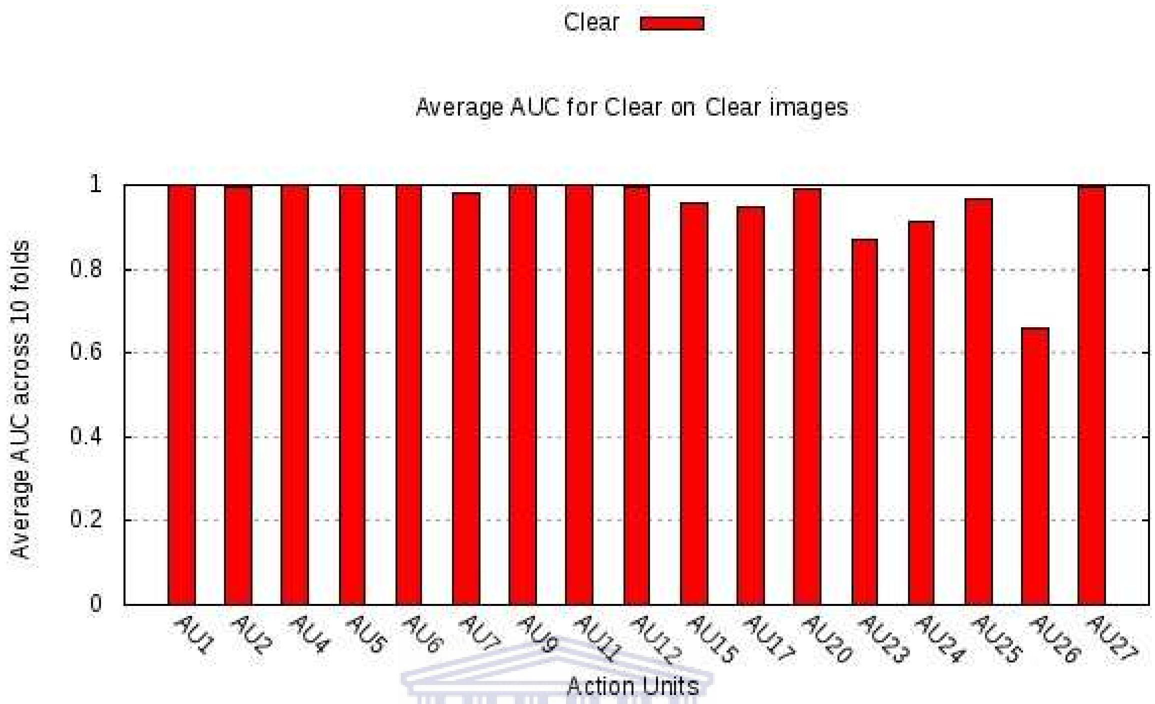


Figure 6.5: Trained using clear, tested on clear images

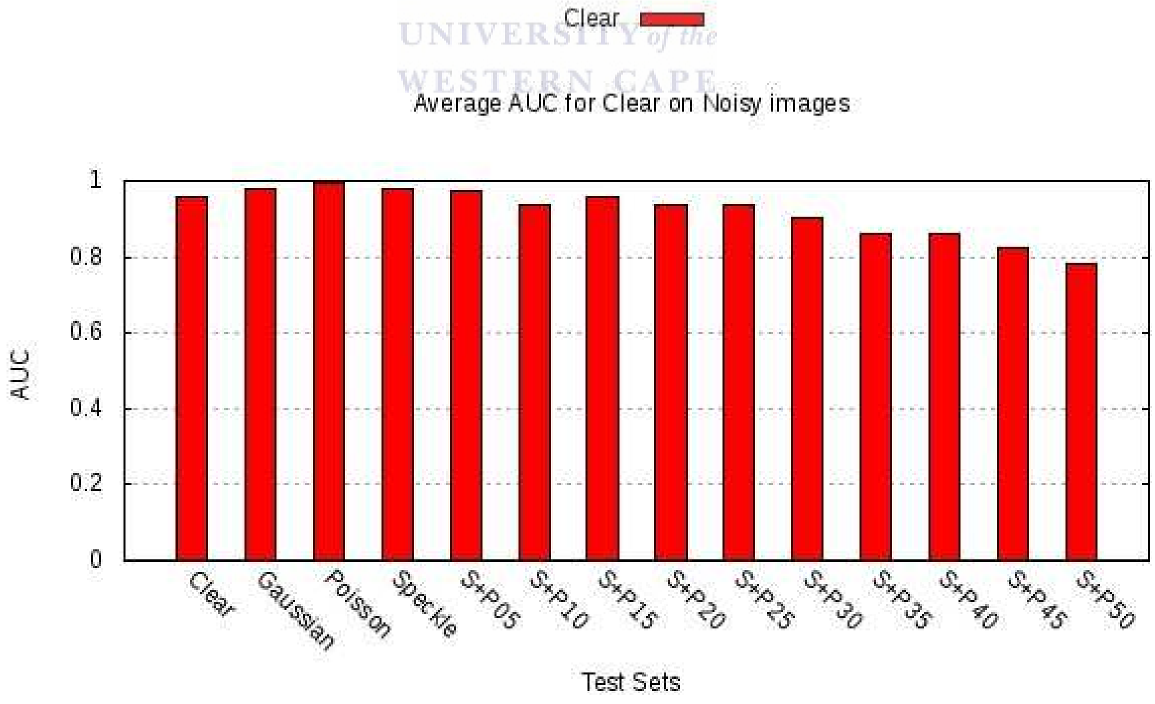


Figure 6.6: Trained Using Clear, Tested on Noisy Images

When training on Clear images and testing against the range of Salt And Pepper images, there is an almost constant decline in accuracy. It should however be noted that even when an SVM is trained with Clear images and tested against Salt And Pepper 50, the lowest AUC values are observed and they still average above 0.75. This shows that FER using Gabor filters and Support Vector machines is quite resilient to noise, particularly in the test sets. If the threshold AUC value for acceptable performance is set to 0.9, the Gabor-SVM system will recognise images with up to 30% Salt & Pepper noise.

The results suggest there might be some degree of bias introduced when images are Gabor filtered. Applying Gaussian, Poisson and speckle noise appears to be having a smoothing effect on images, which is to the favour of the Gabor filter.

6.3.3 Experiment 3: Train on Noisy, Test on Clear

Figure 6.7 shows the average AUC values when the SVMs were trained using noisy images and tested on clear images. The different coloured bars identify the image set used to train SVMs. For instance, the left-most green bar is for the SVM trained with images having Gaussian noise and tested on images having no noise. The horizontal axis shows the image sets on which SVMs were tested, and the vertical axis shows Again it can be seen that the SVM trained using images with Poisson noise added yielded highest AUC values when tested on Clear images – even higher than Clear against Clear. A similar situation was evident in Experiment 2 where Clear images performed best when tested against Poisson images.

6.3.4 Experiment 4: Train on Noisy, Test on Noisy

Table 6.4 shows the average AUC values obtained from each experiment – including Experiment 4. A simplified, graphic representation of the Table 6.4 clarifies the trends in the data, and is shown in Figure 6.8. The medians and standard deviations across all AUs are shown in Figure A.1 and Figure A.2 respectively.

Even though the tables show the full set of results for all training and test sets, the graphs only show the top 5 rows. i.e. the SVM performance when trained with broad category noise - Clear, Gaussian, Poisson, Salt and Pepper, and speckle - and tested on all sets of noise. The reason for this is space related. It is impractical to show all results in one bar graph. Full results for each AU, including the medians and standard deviations, can be found in Appendix B.

Along the bottom axis of Figure 6.8 are the test sets: the sets of images on which SVMs were tested. At the top, as indicated by the legend, are the colours that depict the set of images that were used to train the SVMs. The green bars, for instance, show Clear-SVMs. The left-most green bar, which is in the Clear group, shows the AUC values of Clear-SVMs when tested on Clear images, and averaged across all action units. The exact value this bar depicts can

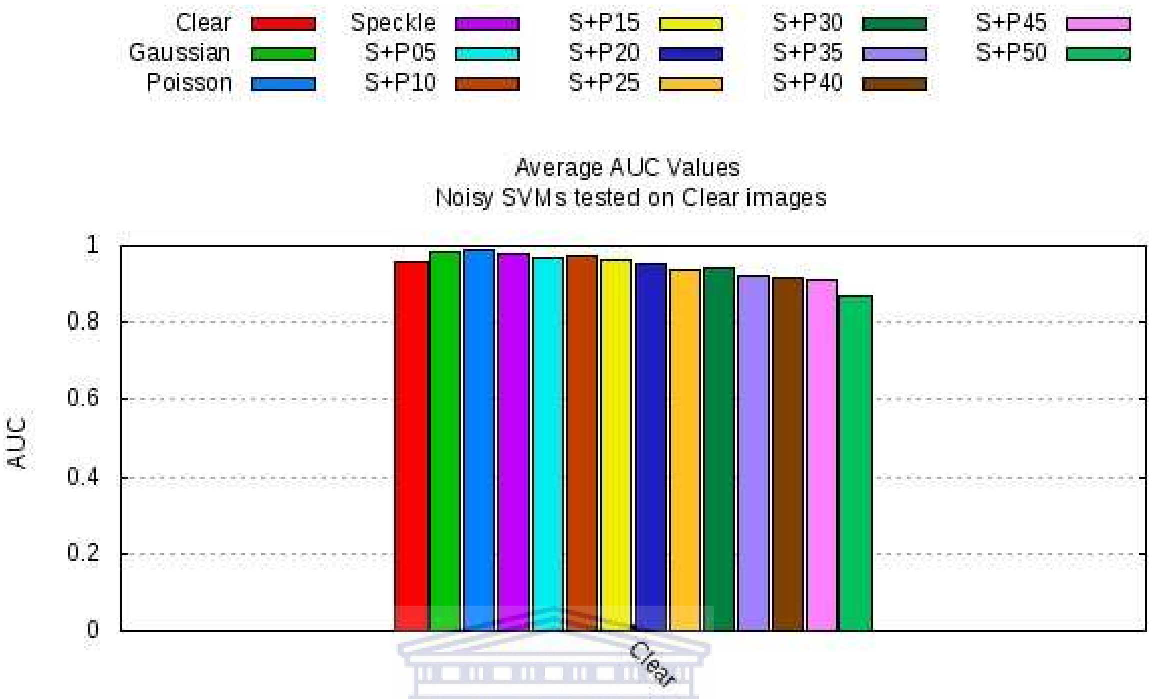


Figure 6.7: Trained Using Noisy, Tested on Clear Images

Table 6.4: Average AUC values across all Action Units

Training Set	Test Sets														
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	**0.957**	**0.978**	**0.993**	**0.979**	**0.976**	**0.936**	**0.959**	**0.937**	**0.937**	**0.905**	*0.862*	*0.861*	*0.822*	0.782	**0.920**
Gaussian	**0.985**	**0.938**	**0.987**	**0.976**	**0.976**	**0.928**	**0.950**	**0.938**	**0.917**	**0.903**	*0.874*	*0.869*	*0.817*	0.798	**0.918**
Poisson	**0.988**	**0.975**	**0.969**	**0.973**	**0.977**	**0.944**	**0.952**	**0.942**	**0.928**	**0.901**	*0.871*	*0.858*	*0.812*	0.780	**0.919**
Speckle	**0.977**	**0.963**	**0.982**	**0.907**	**0.968**	**0.916**	**0.948**	**0.926**	**0.914**	**0.907**	*0.885*	*0.851*	*0.821*	0.787	**0.911**
S+P05	**0.968**	**0.959**	**0.986**	**0.958**	**0.929**	**0.921**	**0.948**	**0.923**	**0.917**	*0.900*	*0.882*	*0.867*	*0.814*	0.797	**0.912**
S+P10	**0.973**	**0.955**	**0.981**	**0.961**	**0.963**	*0.877*	**0.937**	**0.926**	**0.914**	*0.899*	*0.877*	*0.853*	*0.817*	*0.800*	**0.910**
S+P15	**0.961**	**0.961**	**0.963**	**0.950**	**0.961**	**0.915**	*0.861*	**0.914**	*0.892*	*0.884*	*0.863*	*0.848*	*0.815*	0.788	*0.898*
S+P20	**0.954**	**0.956**	**0.967**	**0.946**	**0.947**	**0.913**	**0.923**	*0.850*	*0.893*	*0.879*	*0.878*	*0.839*	*0.817*	0.793	*0.897*
S+P25	**0.935**	**0.958**	**0.961**	**0.948**	**0.949**	**0.916**	**0.919**	*0.889*	*0.846*	*0.871*	*0.884*	*0.855*	*0.821*	0.787	*0.896*
S+P30	**0.941**	**0.930**	**0.941**	**0.932**	**0.941**	*0.886*	**0.920**	*0.865*	*0.893*	*0.804*	*0.855*	*0.820*	*0.807*	0.796	*0.881*
S+P35	**0.921**	**0.925**	**0.941**	**0.934**	**0.926**	*0.878*	**0.909**	*0.877*	*0.846*	*0.859*	*0.823*	*0.820*	*0.818*	0.777	*0.875*
S+P40	**0.914**	**0.917**	**0.903**	**0.912**	**0.921**	*0.867*	*0.899*	*0.879*	*0.850*	*0.846*	*0.863*	0.778	0.778	0.795	*0.866*
S+P45	**0.913**	**0.901**	*0.898*	*0.875*	**0.902**	*0.877*	*0.889*	*0.841*	*0.865*	*0.819*	*0.840*	*0.802*	0.737	0.768	*0.852*
S+P50	*0.871*	*0.869*	*0.894*	*0.877*	*0.887*	*0.870*	*0.860*	*0.833*	*0.855*	*0.805*	*0.815*	*0.805*	0.789	0.699	*0.838*
Averages	**0.947**	**0.942**	**0.955**	**0.938**	**0.945**	**0.903**	**0.920**	*0.896*	*0.890*	*0.870*	*0.862*	*0.838*	*0.806*	0.782	*0.892*

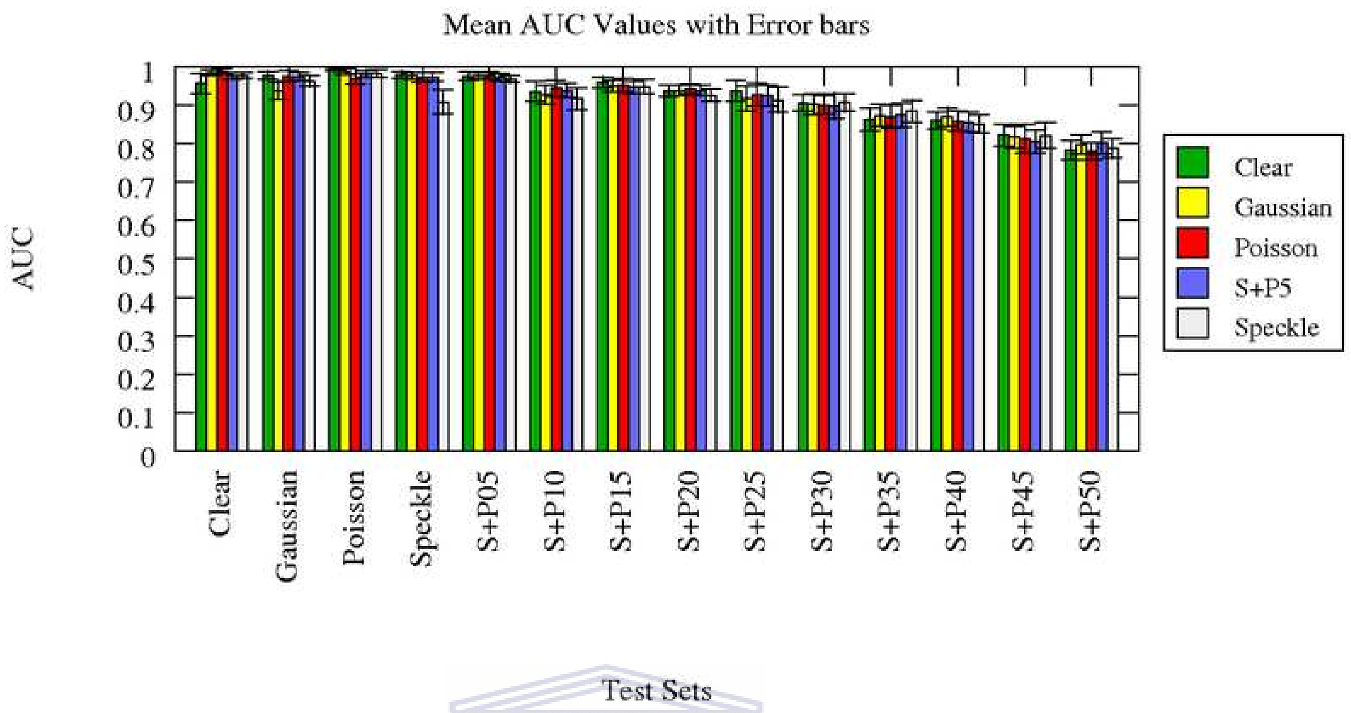


Figure 6.8: Average AUC Values Over All AUs

be found in Table 6.4 and is 0.957.

The cluster-grouping of test sets allows data about general trends in classification to be extracted with ease. For instance, clear-SVMs, Gaussian-SVMs, Poisson-SVMs, salt & pepper-SVMs and speckle-SVMs were able to – with probability greater than 0.9 – correctly classify images that had a salt & pepper noise density of 30% or less.

The bold values in Table 6.4 indicate that the AUC value is above 0.9. The italic values indicate that the AUC value is between 0.9 and 0.8. Anything in this range is considered to be *borderline*. The plain-text values are those less than 0.8 and are *unacceptable*. The results show that a clear region exists in which the borderline AUC values lie. Beyond this arc is where the unacceptable AUC values lie. It is clear that a moderate amount of noise is acceptable and that performance doesn't deteriorate significantly. Images with more than a 25% density of salt and pepper noise, be it in training or in testing, are entering the borderline zone. However, the less noise present in the training set, the more capable the SVM is in recognition of noisy data.

Note that when looking down the diagonal from top left to bottom right, the values for which the training sets are the same as the test sets are observed. For instance, speckle filtered images are used for both training and testing of the SVMs. There is a clear decline in AUC value from top-left to bottom-right for salt & pepper, with the SVMs training using salt and pepper 50 and testing on salt and pepper 50 performing the worst.

Table 6.5: Regression Statistics for clear, Gaussian, Poisson, speckle and salt & pepper 05 SVMs

Statistic	**Clear**	**Gaussian**	**Poisson**	**Speckle**	**S+P05**
R^2	0.905733383	0.913946836	0.879758055	0.879758055	0.819417797
m	-0.003989091	-0.003624242	-0.004129697	-0.003526061	-0.003008485
t	-8.77	-9.22	-10.46	-7.65	-6.03

6.3.5 Regression Analysis on Salt And Pepper Noise

Linear regression was performed on AUC vs salt and pepper noise for the clear-SVMs, Gaussian-SVMs, Poisson-SVMs, speckle-SVMs and salt & pepper-SVMs. Figure 6.9 shows linear regression graphs for each SVM. An R^2 analysis was performed to measure how valid the regression graphs are.

R^2 is the *coefficient of determination* and measures the proportion of the variation in AUC values that is explained by the variation S & P noise. The higher the value of R^2, the better the model fits the data. The *null hypothesis*, $H_0 : \beta = 0$, indicates there is no linear relation between AUC and salt and pepper noise for clear, Gaussian, Poisson, speckle and salt & pepper 05 SVMs. The *alternate hypothesis*, $H_1 : \beta \neq 0$, indicates there is a relation between AUC and salt and pepper noise for the SVMs.

Table 6.5 shows some statistics relevant to the R^2 tests. m is the gradient of the linear model $f(x) = c + mx$. t refers to the test statistic. There are 8 degrees of freedom, and with a significance level of 99% ($alpha = 1\%$), the rejection region is where $t > t_{\alpha/2,n-2} = 3.355$ or $t < -t_{\alpha/2,n-2} = -3.355$.

From Table 6.5, the clear-SVM has a $R^2 = 90.57\%$. This means that 90.57% of the variation in AUC values are explained by the variation in the salt and pepper noise. The remaining 9.43% is unexplained. The coefficient of determination does not have a critical value from which conclusions can be deduced.

For all SVMs in Table 6.5, the t-values are $\ll -3.355$, i.e there is overwhelming evidence that a linear relationship exists between the AUC values produced by each SVM and the amount of salt and pepper noise they were tested on.

6.4 Discussion

This chapter has answered some important questions in the field of FER. In this section, the results supporting each question will be summarised and an answer to each question will be discussed.

How is recognition affected when noise is introduced in Gabor+SVM FER?

The results show that SVMs trained with Gabor filters exhibit unquestionable degradation in recognition rate when tested on images with increasing levels of noise. There is no doubt at all that recognition rates drop to lower recognition

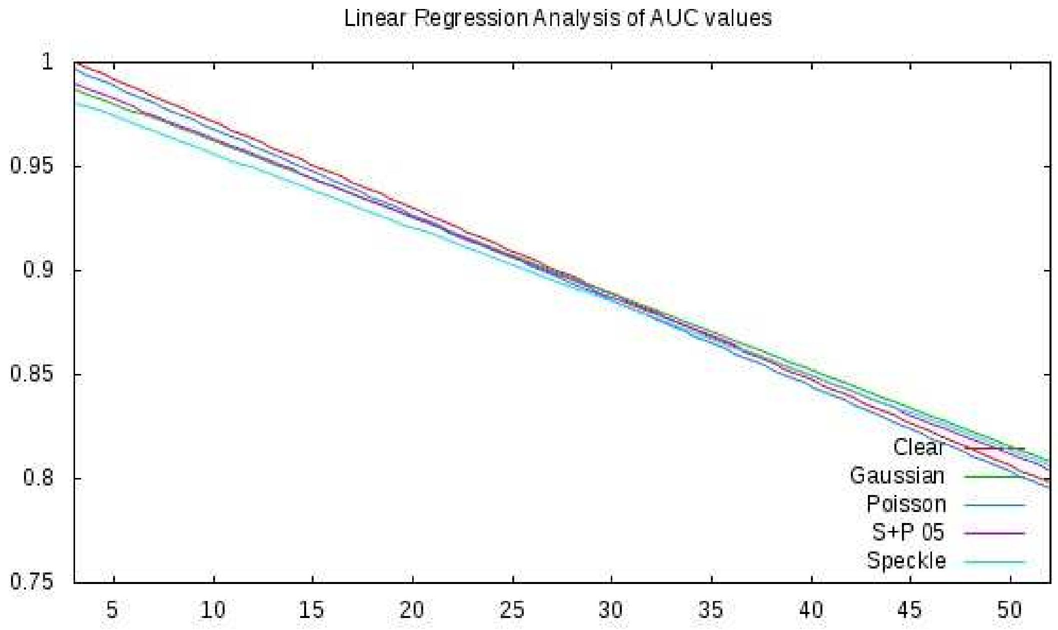


Figure 6.9: Linear regression curves showing AUC values of Clear-, Gaussian-, Poisson-, salt and pepper- and speckle-SVMs dropping as the salt and pepper noise increases.

rates where the noise density increases. This was shown clearly in Figure 6.9. Further, the rate at which recognition drops is only slightly different between SVM training sets.

At what stage does recognition become unacceptable?

The results show that an SVM is able to - with probability greater than 0.9 - correctly classify images that have a noise density of as much as 30% salt & pepper. If the certainty constraint is relaxed to, say, AUC = 0.8, then the system can cope with Salt & Pepper density levels as high as 45%. It can be seen that the AUC values of salt & pepper 50-SVMs do not yield any AUC values above 0.9.

Is it possible to improve recognition rates for noisy environments?

The results suggest that some types of noise, specifically Gaussian and Poisson, do in fact boost recognition to a level greater than when solely using clear images. This is an unexpected outcome and deserves further investigation. The experiments hint that training SVMs with both clear and some combination of noisy data will lead to better, more robust classification performance.

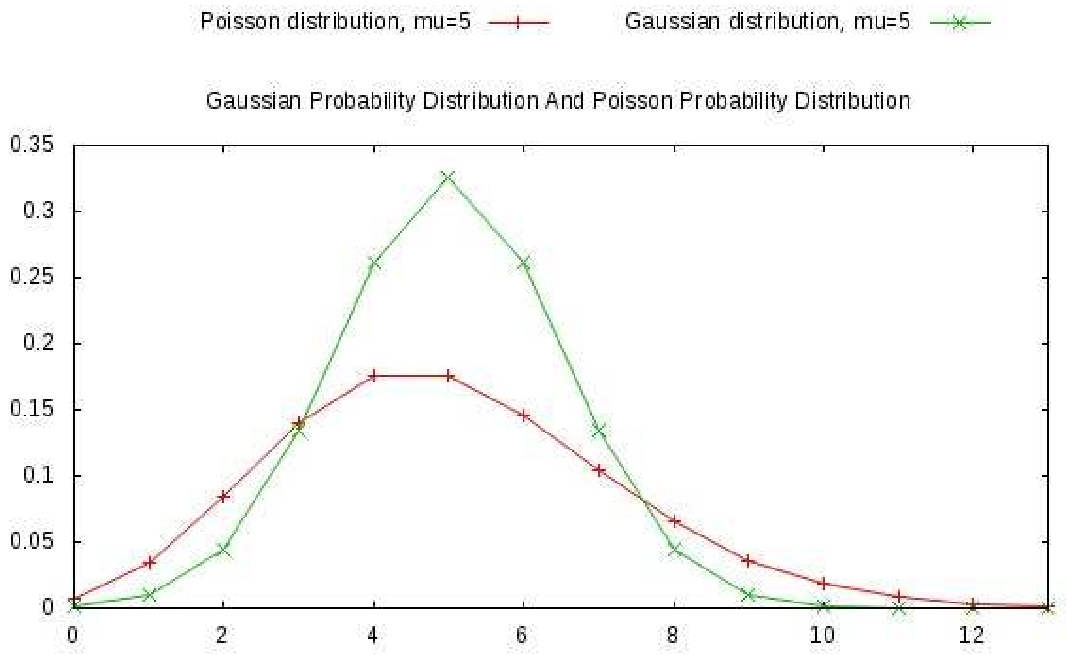


Figure 6.10: The Gaussian and Poisson Probability Distributions. Both graphs are similar to the Gaussian Envelope component of a Gabor filter (see Sec. 5.2).

Does FER with Gabor filters and SVMs degrade gracefully?

The graphs in Figure 6.9 indicate that degradation is constant. It is also clear, from Table 6.4, that all Noisy-SVMs degrade in performance as the amount of noise in test images increases, and that they all do so at similar gradients.

Hence, for the Salt & Pepper 50 SVM, even though none of the AUC values are above 0.9, the AUC values still degrade at the same rate as noise increases, as that of other SVMs.

Why do Gaussian- and Poisson-SVMs outperform Clear-SVMs?

An in-depth solution to this question could serve as an avenue for further investigation. It is well documented that Gabor filters smooth out noise in signals. Chao et al. [11] present a fingerprint recognition paper that suggests that if the standard deviation of the Gaussian envelope of a Gabor filter is large enough, the filter will be more robust to noise, but tiny details of the face will be lost as a result of smoothing. If the standard deviation is too small, the Gabor filter will not be effective in eliminating the noise. This explanation is independantly corroborated by both Krishan [44] and Rosshidi and Hadi [65]. Some conclusions from literature that are in line with the observations from experiments in this chapter are:

- Gabor filters produce good results when applied to images with Gaussian noise,

- All tested variances perform well when applied to images with Poisson noise,
- For images with speckle noise, Gabor filters with low variance produce the best results.

Bhuiyan and Liu [6] optimize a Gabor filter for use with a neural network for face recognition. They use their Gabor filters to remove both salt and pepper noise and Gaussian noise and conclude that their Gabor filter effectively reduces both types of noise. Figure 6.10 shows that Poisson graph can easily be moulded to appear like the Gaussian graph by tweaking its parameter. When the variance of these distributions are aligned to that of the Gabor filter, the effect will be similar.

The most likely explanation as to why images with Gaussian and Poisson noise have higher classification rates than Clear images, is that the artificially applied Gaussian and Poisson noise that was applied to the Clear images smoothes out the noise. This appears to assist the FER process.

6.5 Summary

This chapter presented a detailed description of the experimental procedure, as well as of the four experiments performed:

1. A model was created that set a benchmark against which to compare the classification performance of SVMs trained or tested with noisy images. Clear, untainted images were used in both training and testing the SVMs for each AU.
2. SVMs were trained using clear images and then tested on images with noise.
3. The ability of SVMs to extrapolate was tested by training SVMs with noisy images, and then testing on clear images.
4. SVMs were trained on noisy images, and then tested on all types of noisy images.

The results of the experiments were presented and analysed. In order to determine trends in the results, a regression analysis was then performed on the Salt And Pepper tests. The chapter ended with a discussion of the results and of the regression analysis.

Chapter 7

Conclusions And Directions for Further Research

This chapter concludes the thesis by listing the contributions made by this work. Suggestions for future work are then provided.

7.1 Contributions

First, an existing facial expression recognition framework was used to generate a results base against which to compare new results. Using A' as a metric upon which to compare recognition accuracy of the FER system, models were generated that describe the generalization properties of the FER system. Across 17 action units, when trained using clear images and tested on clear images, this framework was weighted by AU prevalence and produced a mean AUC value of 0.947 and a median of 0.988.

Next, various noise models were added to the facial images. These noisy image sets were used in the FER framework in various combinations of clear and noisy training and test images. The noise models used were: Gaussian noise, Poisson noise, salt And pepper noise, and speckle noise. The mean AUC results for an SVM trained using Gabor-filtered clear images and tested on Gaussian, Poisson, salt & pepper and speckle noise images are 0.978, 0.993, 0.976 and 0.979 respectively.

Images from these sets of Salt & Pepper noise were used to train SVMs, as well as test the SVMs. Images from the other noise distributions – Clear, Gaussian, Poisson and Speckle – were also used in training SVMs and tested on this scale of Salt & Pepper noise, and vice versa.

To obtain a comprehensive set of results, the effects of training SVMs with noisy images and then testing on the same type of noisy images, were also examined. These noisy-SVMs were cross tested on test sets with all other types of noise, as well as on clear images. The result was a 14X14 grid of AUC values describing Gabor+SVM recognition rates over different sets of clear and noisy images.

All in all, there were 14 sets of images:

- Clear;
- Clear with Gaussian noise;
- Clear with Poisson;
- Clear with Speckle;
- Clear with Salt & Pepper of density 05;
- Clear with Salt & Pepper of density 10;
- Clear with Salt & Pepper of density 15;
- Clear with Salt & Pepper of density 20;
- Clear with Salt & Pepper of density 25;
- Clear with Salt & Pepper of density 30;
- Clear with Salt & Pepper of density 35;
- Clear with Salt & Pepper of density 40;
- Clear with Salt & Pepper of density 45;
- Clear with Salt & Pepper of density 50.

All of these sets of images were used to train a Gabor+SVM system, which was then tested on every other set of images. The average of all SVMs together was 0.897.

A predictive model was generated from a scale of Salt & Pepper noise that started with a density of 5% and increased in 5 unit intervals to 50%. From the experiments using the scale of Salt & Pepper noise, a model was created that could be used to predict what happens when the density of Salt & Pepper noise increases. The model was found to follow an linear graph in which the AUC decreases gracefully as the noise density increases.

Finally, the results obtained when testing a clear-SVM on Gaussian images or Poisson images corroborate what has been shown in independent research: that Gabor filters remove noise when the variance of the Gabor's Gaussian envelope is similar to that of the Gaussian-like noise. It was also shown that applying noise to clear images, and then applying a Gabor filter will increase the recognition rate of a Gabor+SVM system.

7.2 Directions for Future Work

The directions for future work are three-fold. Each will be dealt with individually.

7.2.1 Noise in Classification

It was seen that the combination of Gabor filters and support vector machines work well together for the task of facial expression recognition. However, it is unclear which component, Gabor filters or support vector machines or both together, plays the major role in recognizing that the clear images are in fact the same as the noisy images after noise has been added. It is also not known whether this is specific to FER. It would be interesting to know for certain if support vector machines are robust enough to withstand the effects of noise in object classification in general or whether it is the Gabor filtering process that removes the effects of noise and how much noise it removes.

7.2.2 Using Gaussian and Poisson noise in Training

The results in Chapter 6 suggest that training SVMs using images that have Gaussian or Poisson noise improve recognition rates. Would it be useful to construct a training set that consists of clear images as well as images having some combination of Gaussian or Poisson noise. This needs to be investigated further.

7.2.3 Moving out of the Laboratory

With the advent of security technology, mobile technology and ubiquitous computing, both low and high quality means of recording images and video are infiltrating the markets. Research needs to move out of the laboratory and into the real world, using sub-optimal photographic situations and technology such as CCTV cameras, cellular mobile phones and webcams. In this way it would be able to deduce the feasibility of spontaneous facial expression recognition as it occurs in commonplace society.

Appendix A

Results: AUC Statistics

Appendix A shows two statistics for measuring the results obtained for the Action Units tested: the median and standard deviation. The tables were generated using the data in Appendix B.

The median of an ordered set of values is the value in the middle that separates the top half of those values from the bottom half. The standard deviation is an indicator of the spread of values from the mean. If there is high spread, then there is a large range of values. In this context, a high spread could mean one of problems: the variability in facial data is not consistent throughout the set, or, the classifier could not generalise well enough. However, it is important to note that one statistic cannot be considered without also reading the other.

Table A.1 contains the averages of medians of AUC values derived from corresponding ROC curves. Table A.2 contains the averages of standard deviations of AUC values as read from corresponding ROC curves. Every value in the table is the average of 10 folds. Figures A.1 and A.2 are graphical representations of the data in Tables A.1 and A.2 respectively.

Table A.1: AU=Medians

Training Set	Test Sets														
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	**0.988**	**0.986**	**1.000**	**0.983**	**0.985**	**0.934**	**0.970**	**0.944**	**0.947**	*0.891*	*0.851*	*0.861*	0.792	0.762	**0.921**
Gaussian	**0.994**	**0.956**	**0.997**	**0.984**	**0.978**	**0.940**	**0.966**	**0.934**	**0.932**	*0.882*	*0.873*	*0.868*	*0.804*	0.787	**0.921**
Poisson	**0.997**	**0.990**	**0.988**	**0.985**	**0.980**	**0.938**	**0.964**	**0.943**	**0.942**	*0.889*	*0.850*	*0.869*	*0.827*	0.758	**0.923**
Speckle	**0.988**	**0.978**	**0.994**	**0.938**	**0.976**	*0.900*	**0.961**	**0.926**	**0.931**	**0.910**	*0.890*	*0.854*	*0.840*	0.764	**0.918**
S+P05	**0.985**	**0.973**	**0.995**	**0.972**	**0.940**	**0.915**	**0.961**	**0.921**	**0.914**	*0.882*	*0.853*	*0.868*	0.797	0.772	**0.911**
S+P10	**0.974**	**0.965**	**0.984**	**0.963**	**0.971**	*0.889*	**0.936**	**0.910**	**0.911**	*0.864*	*0.879*	*0.837*	*0.824*	0.770	**0.905**
S+P15	**0.965**	**0.967**	**0.975**	**0.957**	**0.969**	**0.920**	*0.858*	**0.911**	**0.904**	*0.871*	*0.849*	*0.847*	*0.833*	0.783	**0.901**
S+P20	**0.963**	**0.961**	**0.980**	**0.939**	**0.959**	**0.912**	**0.941**	*0.861*	**0.908**	*0.873*	*0.846*	*0.846*	*0.816*	0.778	*0.899*
S+P25	**0.935**	**0.955**	**0.971**	**0.964**	**0.948**	**0.933**	**0.949**	*0.870*	*0.863*	*0.881*	*0.854*	*0.833*	*0.833*	0.768	*0.897*
S+P30	**0.933**	**0.930**	**0.943**	**0.957**	**0.924**	*0.894*	**0.927**	*0.872*	*0.893*	*0.804*	*0.842*	*0.812*	0.792	*0.804*	*0.880*
S+P35	**0.911**	**0.928**	**0.929**	**0.943**	**0.930**	*0.869*	**0.903**	*0.863*	*0.854*	*0.835*	*0.817*	*0.835*	*0.823*	0.758	*0.871*
S+P40	**0.905**	**0.901**	**0.902**	**0.919**	**0.933**	*0.880*	*0.900*	*0.870*	*0.862*	*0.845*	*0.839*	0.774	0.781	*0.805*	*0.865*
S+P45	*0.887*	*0.895*	**0.914**	*0.896*	*0.888*	**0.909**	*0.889*	*0.819*	*0.879*	*0.821*	*0.818*	0.770	0.722	0.735	*0.846*
S+P50	*0.857*	*0.864*	**0.903**	*0.885*	*0.886*	*0.864*	*0.866*	*0.825*	*0.855*	*0.810*	0.786	*0.826*	0.765	0.701	*0.835*
Averages	**0.949**	**0.946**	**0.962**	**0.949**	**0.948**	**0.907**	**0.928**	*0.891*	*0.900*	*0.861*	*0.846*	*0.836*	*0.804*	0.767	*0.892*

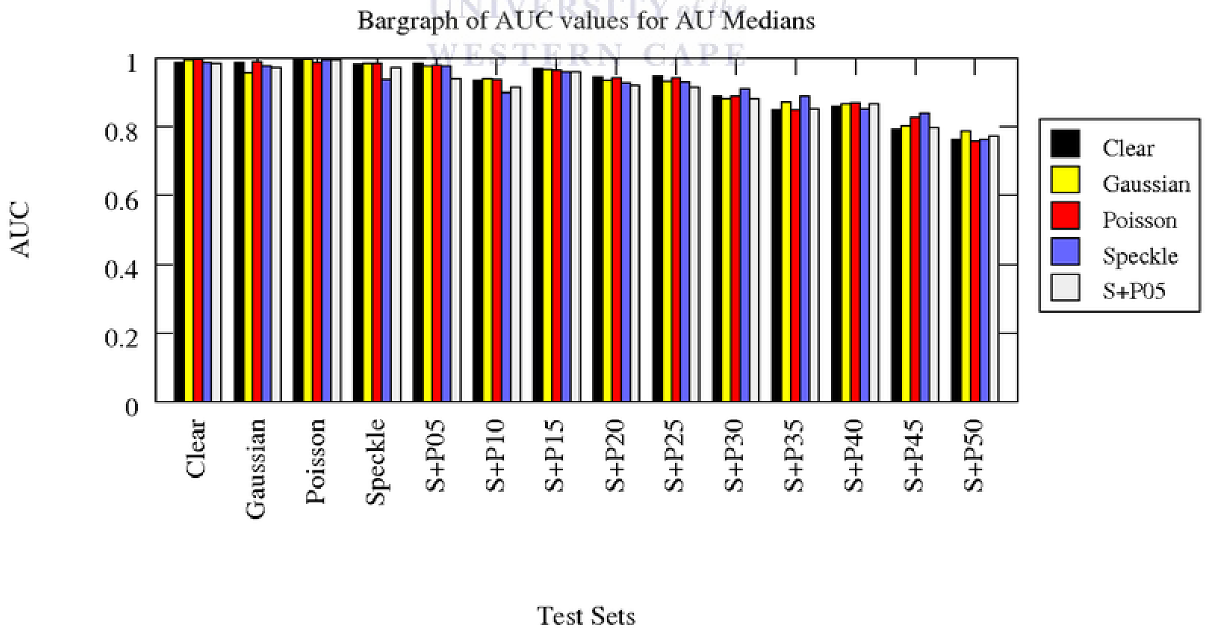


Figure A.1: Medians of AUC Values Over All AUs

Table A.2: AU=Standard Deviation

Training Set	Test Sets														
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	0.084	0.031	0.013	0.028	0.034	0.082	0.046	0.048	0.085	0.066	0.093	0.072	0.089	0.080	0.061
Gaussian	0.021	0.072	0.024	0.026	0.024	0.076	0.044	0.042	0.103	0.084	0.090	0.077	0.090	0.079	0.061
Poisson	0.019	0.044	0.045	0.041	0.029	0.069	0.057	0.038	0.095	0.074	0.093	0.081	0.117	0.073	0.063
Speckle	0.024	0.046	0.029	0.098	0.029	0.093	0.056	0.051	0.102	0.072	0.092	0.074	0.108	0.081	0.068
S+P05	0.050	0.061	0.020	0.051	0.078	0.091	0.055	0.053	0.087	0.072	0.070	0.081	0.101	0.079	0.068
S+P10	0.026	0.052	0.023	0.042	0.042	0.121	0.057	0.047	0.089	0.069	0.091	0.082	0.100	0.079	0.066
S+P15	0.040	0.054	0.048	0.046	0.036	0.081	0.130	0.058	0.136	0.085	0.090	0.075	0.105	0.081	0.076
S+P20	0.041	0.048	0.036	0.046	0.057	0.065	0.069	0.098	0.109	0.066	0.069	0.087	0.083	0.081	0.068
S+P25	0.068	0.037	0.041	0.055	0.044	0.087	0.076	0.070	0.134	0.089	0.080	0.083	0.118	0.080	0.076
S+P30	0.041	0.082	0.055	0.075	0.052	0.095	0.064	0.101	0.098	0.153	0.118	0.096	0.088	0.102	0.087
S+P35	0.050	0.062	0.058	0.064	0.053	0.091	0.072	0.056	0.148	0.105	0.109	0.118	0.085	0.085	0.083
S+P40	0.063	0.064	0.088	0.067	0.070	0.130	0.074	0.069	0.134	0.121	0.079	0.114	0.098	0.088	0.090
S+P45	0.069	0.084	0.073	0.106	0.070	0.133	0.072	0.094	0.112	0.130	0.086	0.105	0.123	0.104	0.097
S+P50	0.133	0.103	0.061	0.099	0.091	0.064	0.109	0.082	0.104	0.126	0.093	0.119	0.109	0.145	0.103
Averages	0.052	0.060	0.044	0.060	0.051	0.091	0.070	0.065	0.110	0.094	0.089	0.090	0.101	0.088	0.076

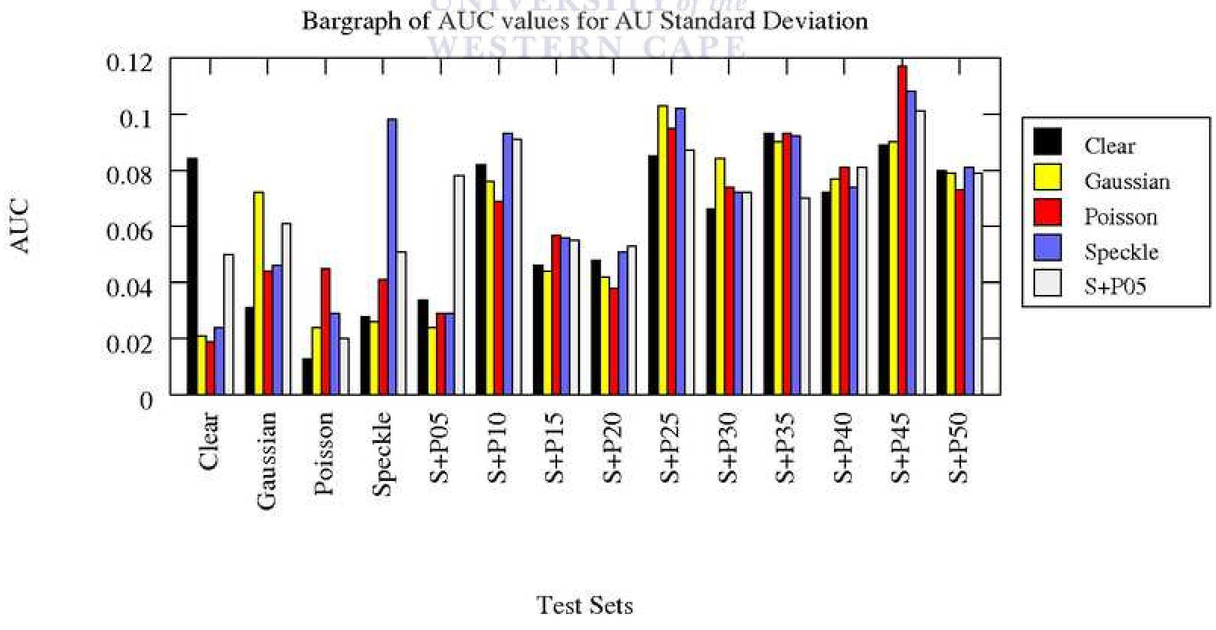


Figure A.2: Standard Deviations in AUC Values Over All AUs

Appendix B

Results: Individual Results Per AU

Appendix B presents the cross-training and testing tables and graphs for each action unit. In all experiments, 10-fold cross testing was performed, each testing yielding one ROC curve and thus one AUC value. Each action unit has one table, which contains the AUC values derived from corresponding ROC curves and averaged over the number of folds. For completion, every filter was cross trained with every other filter. All graphs are presented first, followed by all of the tables.

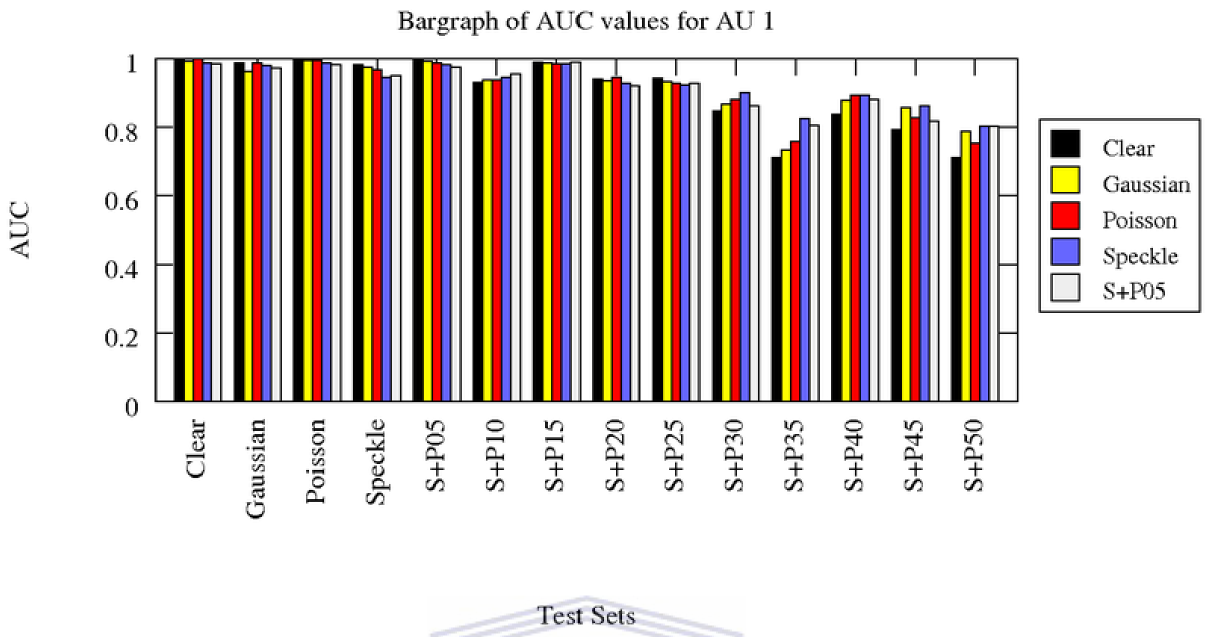


Figure B.1: Bargraph of AUC values for AU 1

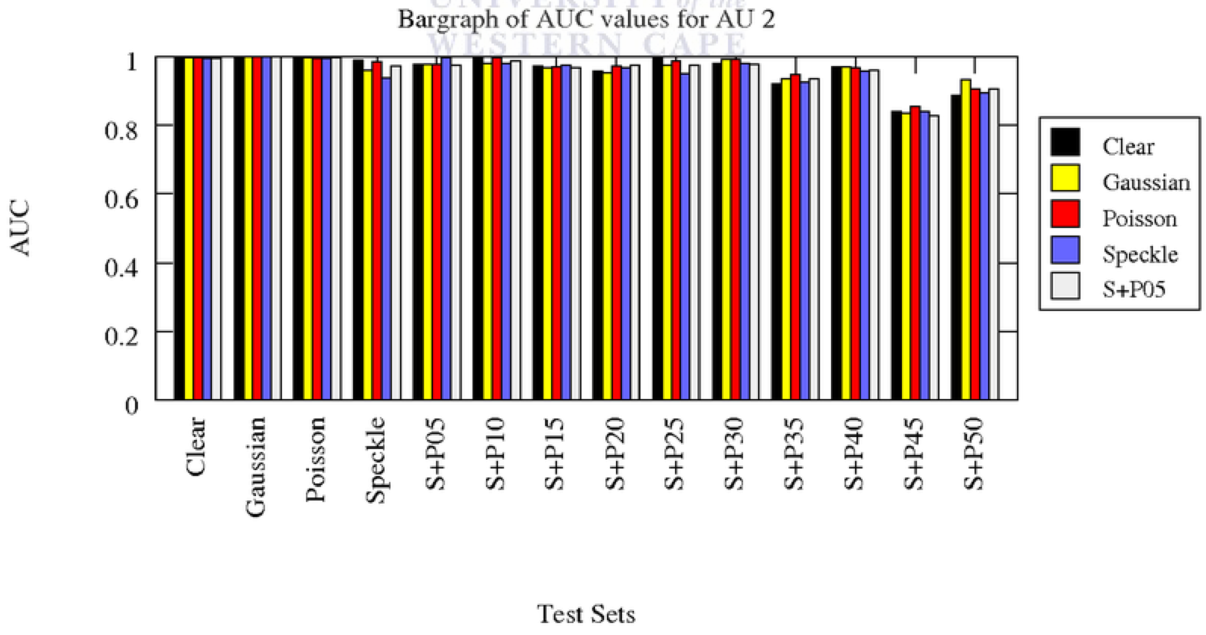


Figure B.2: Bargraph of AUC values for AU 2

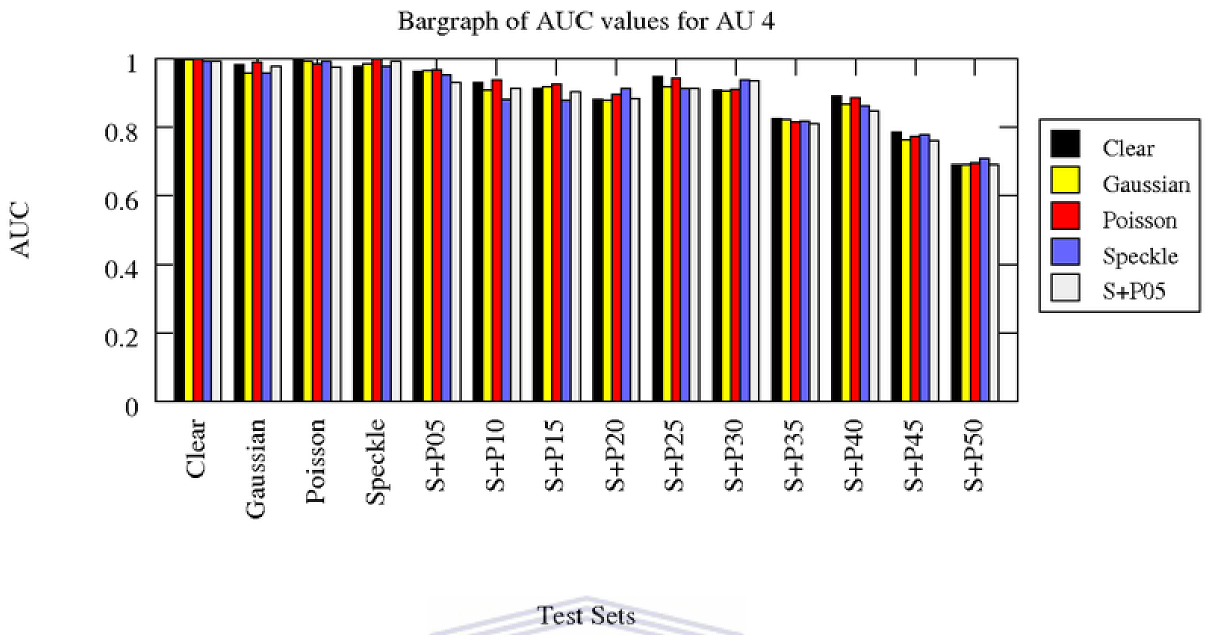


Figure B.3: Bargraph of AUC values for AU 4

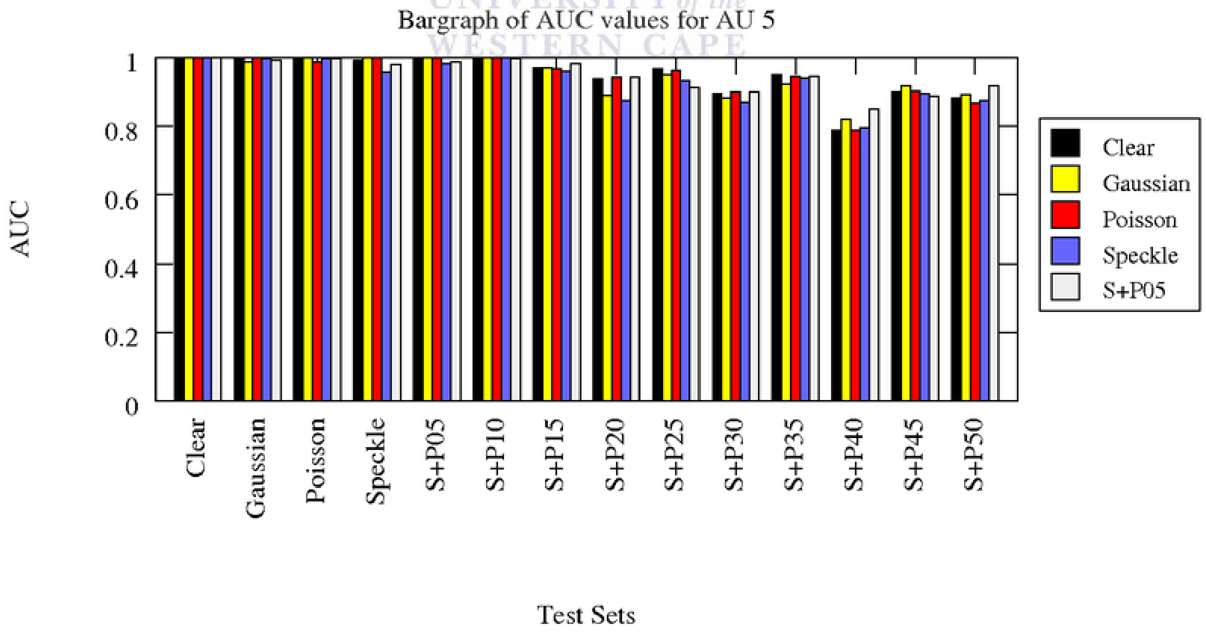


Figure B.4: Bargraph of AUC values for AU 5

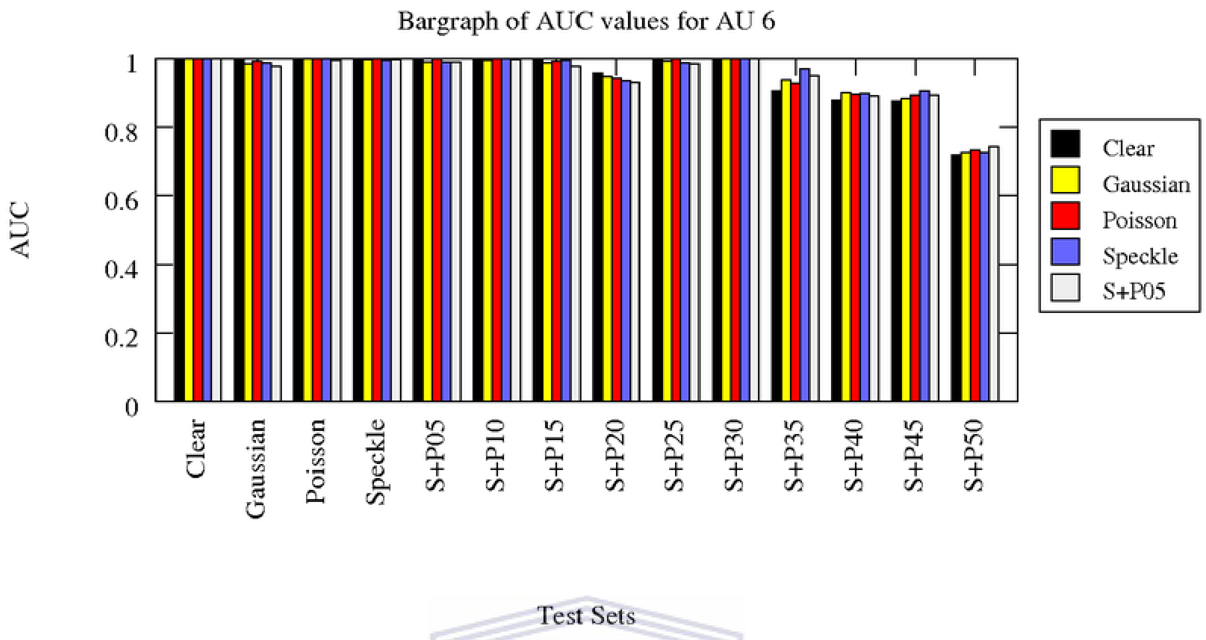


Figure B.5: Bargraph of AUC values for AU 6

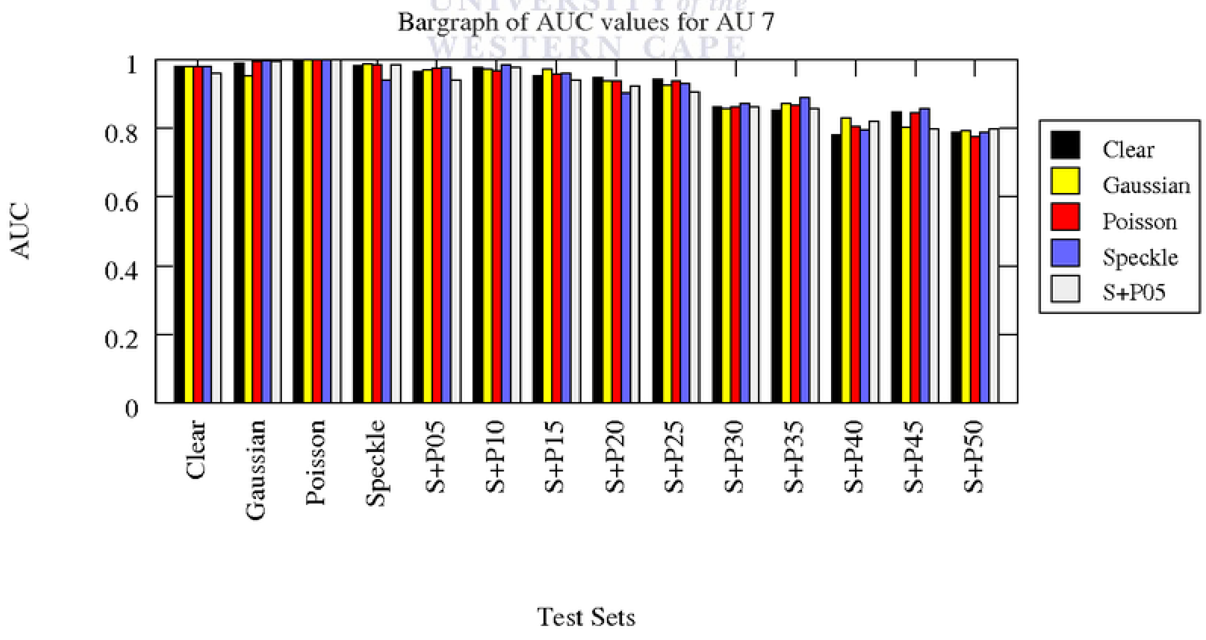


Figure B.6: Bargraph of AUC values for AU 7

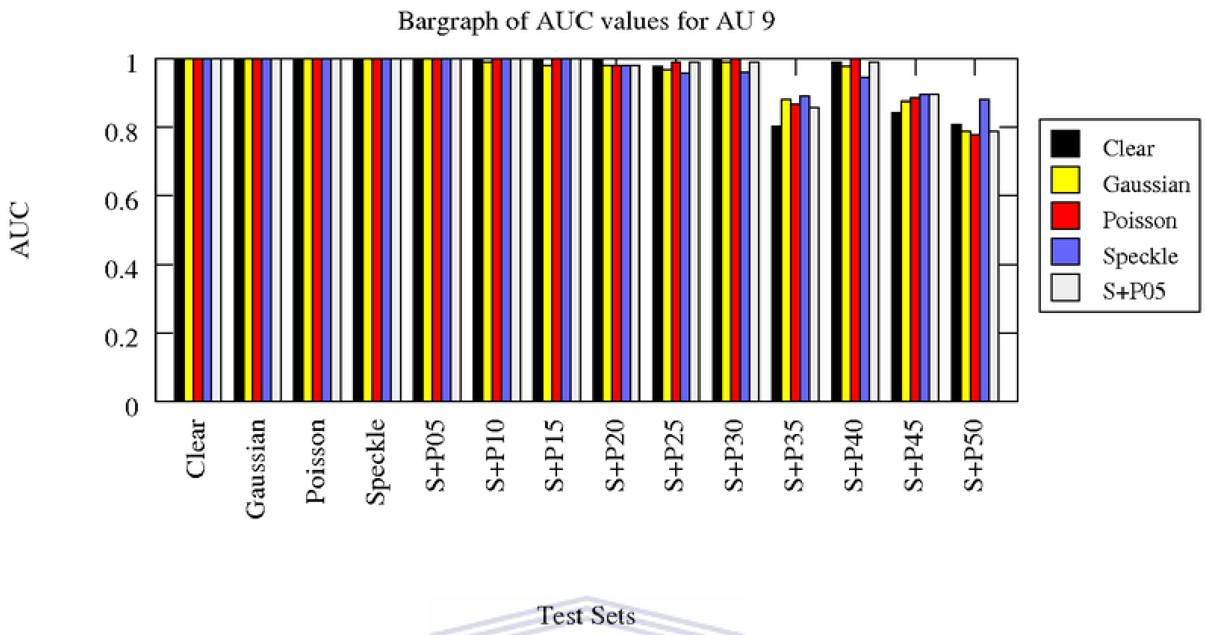


Figure B.7: Bargraph of AUC values for AU 9

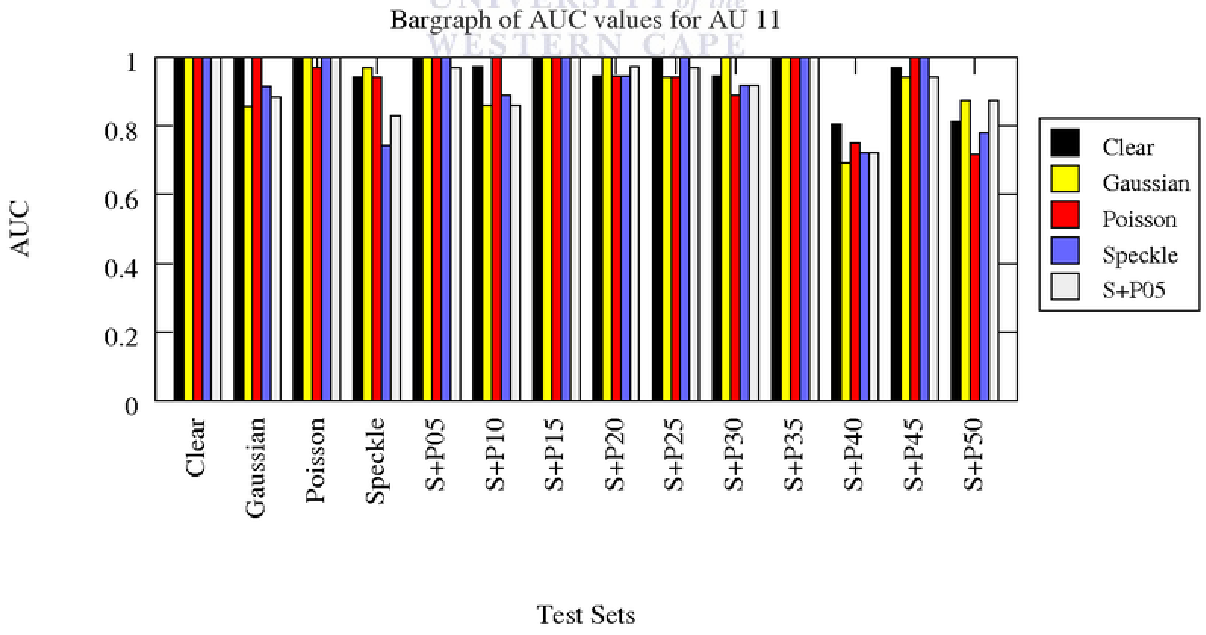


Figure B.8: Bargraph of AUC values for AU 11

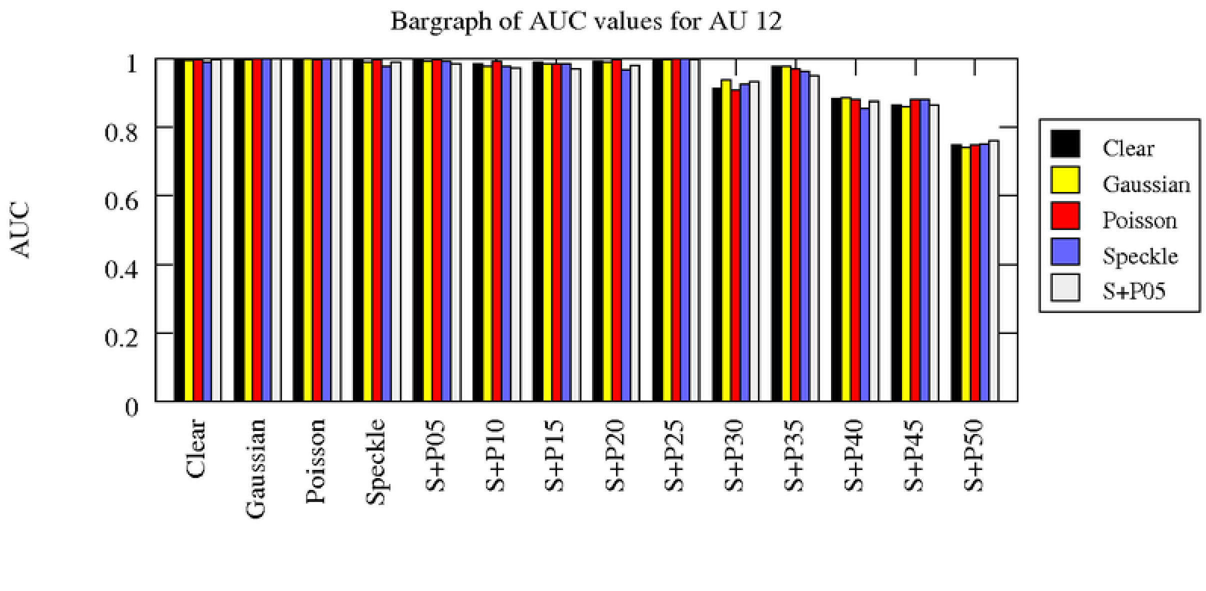


Figure B.9: Bargraph of AUC values for AU 12

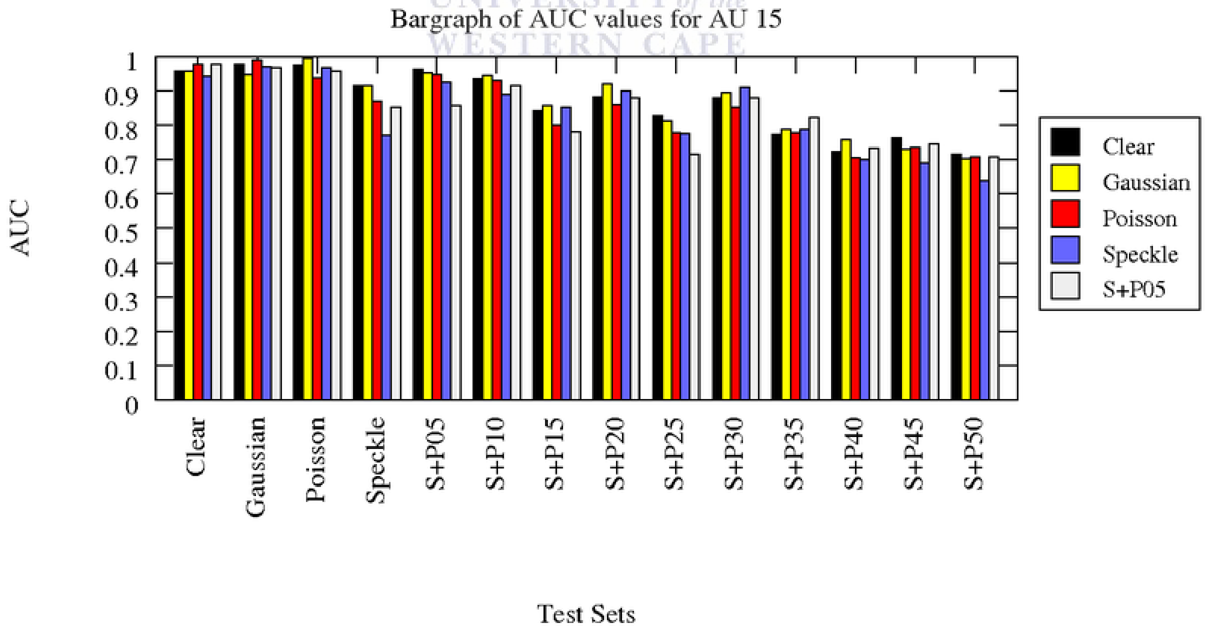


Figure B.10: Bargraph of AUC values for AU 15

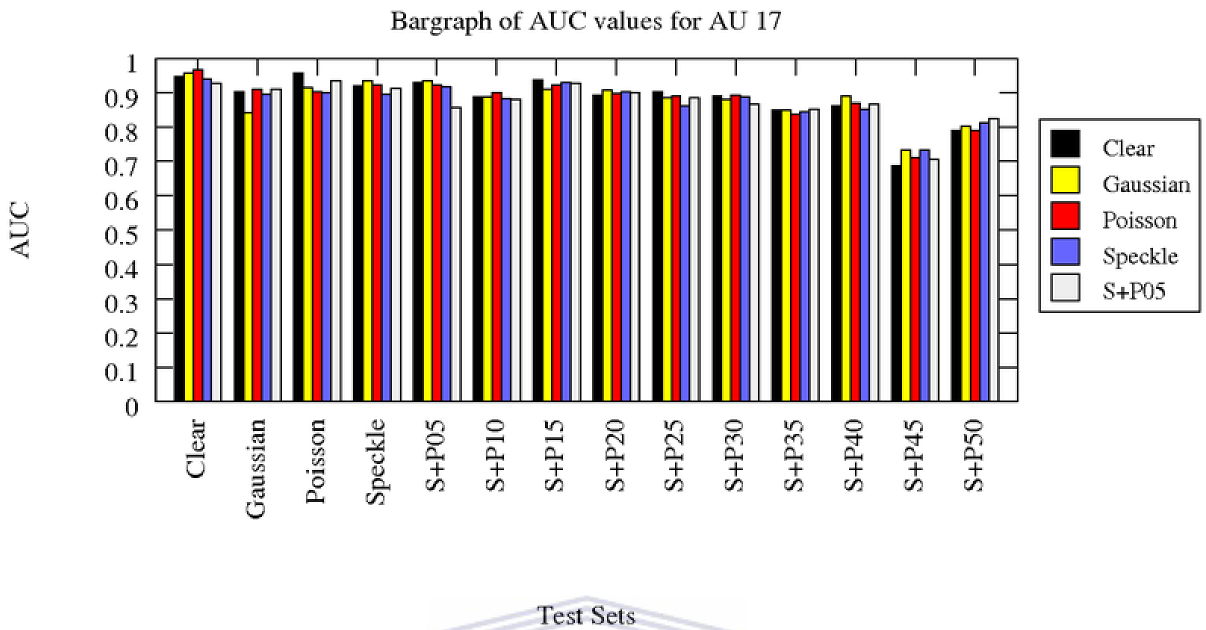


Figure B.11: Bargraph of AUC values for AU 17

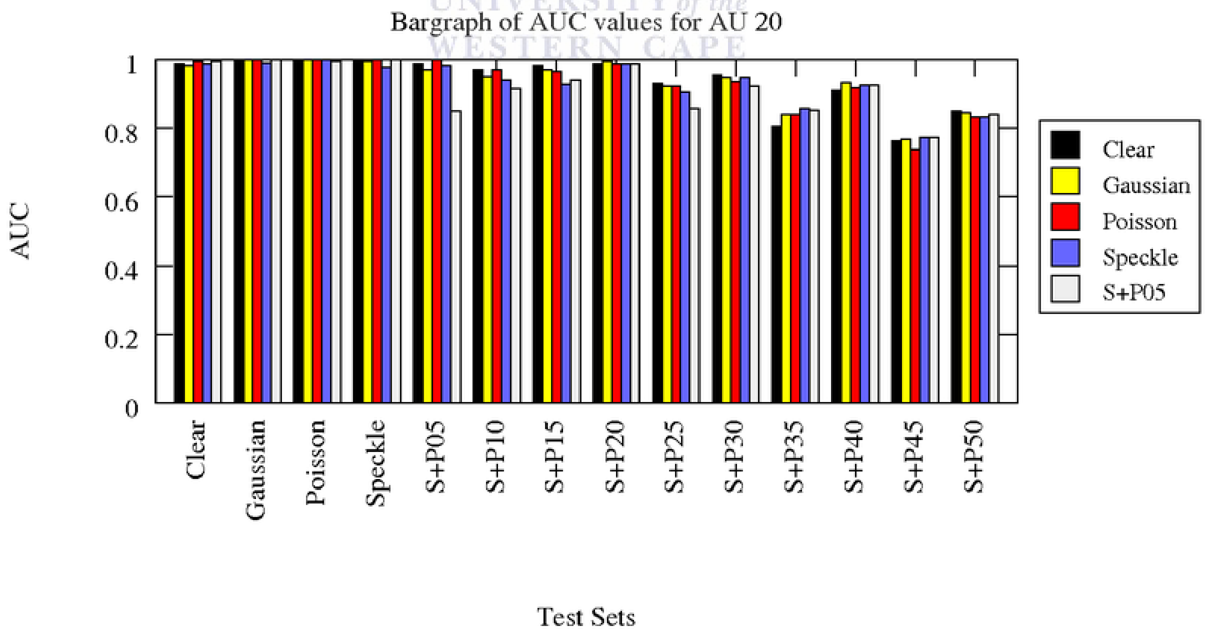


Figure B.12: Bargraph of AUC values for AU 20

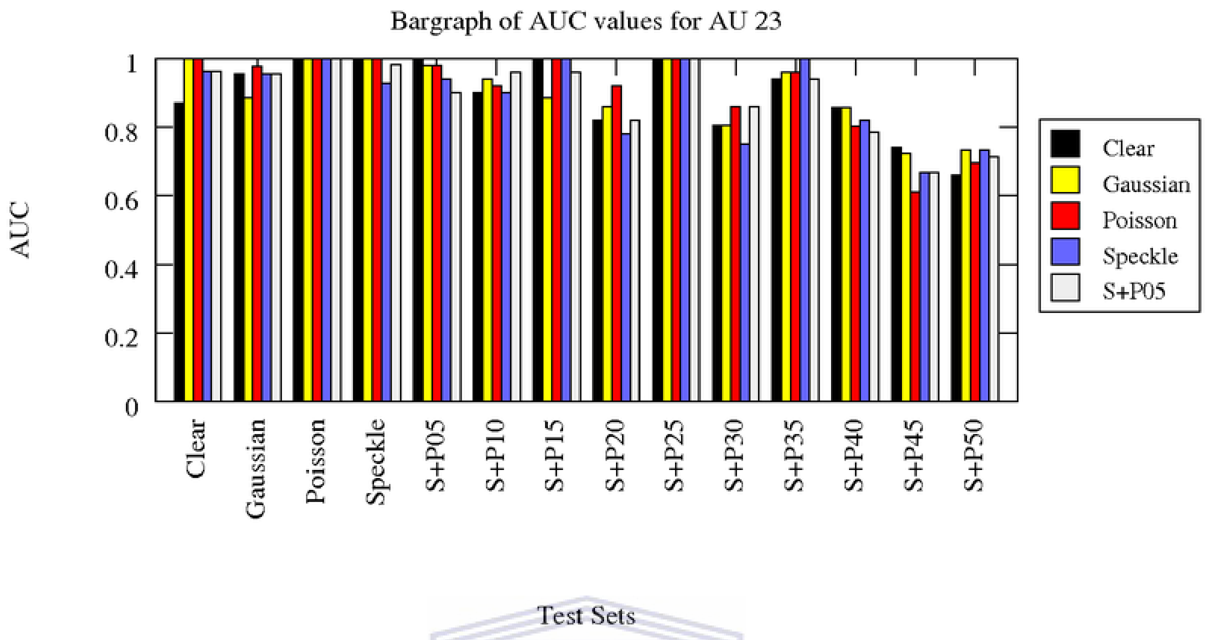


Figure B.13: Bargraph of AUC values for AU 23

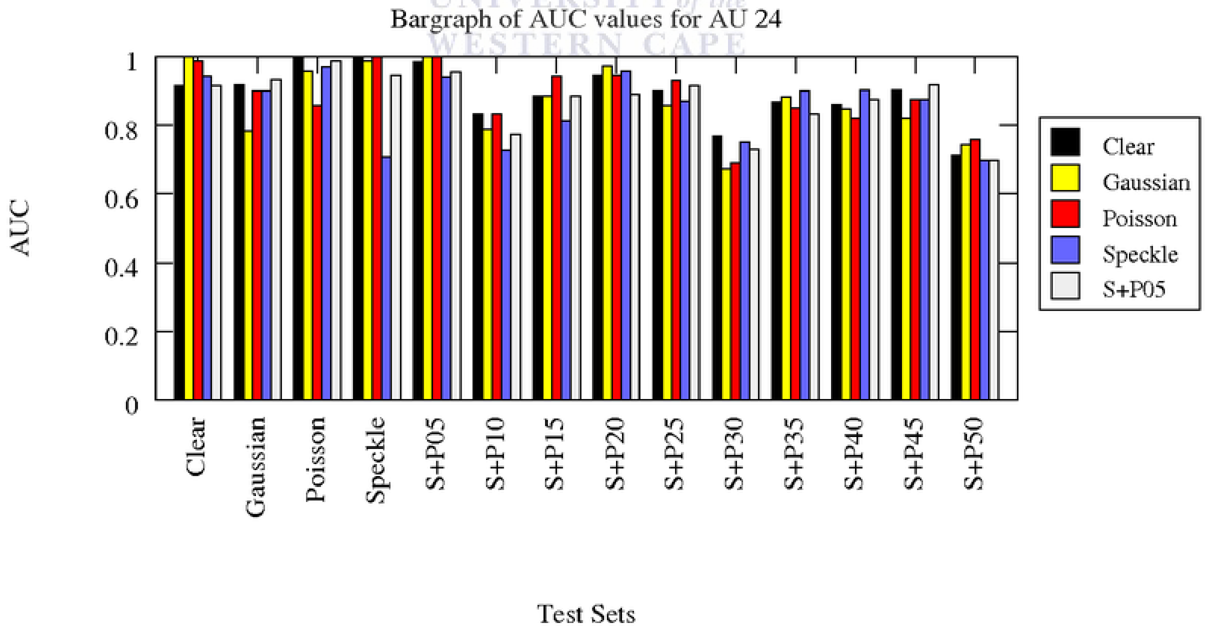


Figure B.14: Bargraph of AUC values for AU 24

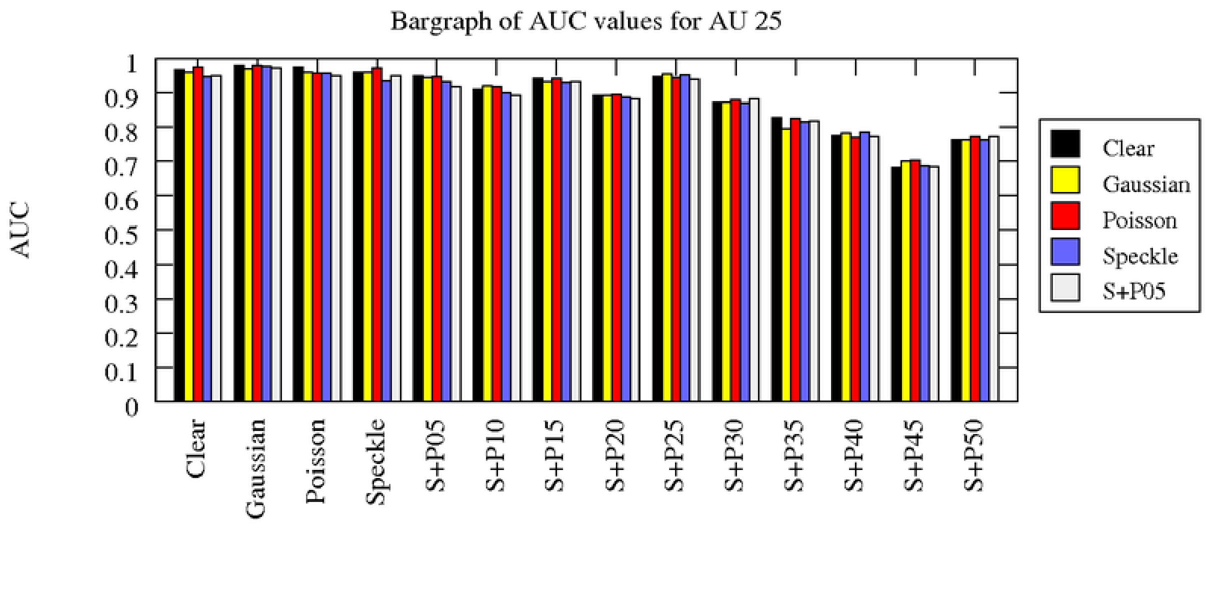


Figure B.15: Bargraph of AUC values for AU 25

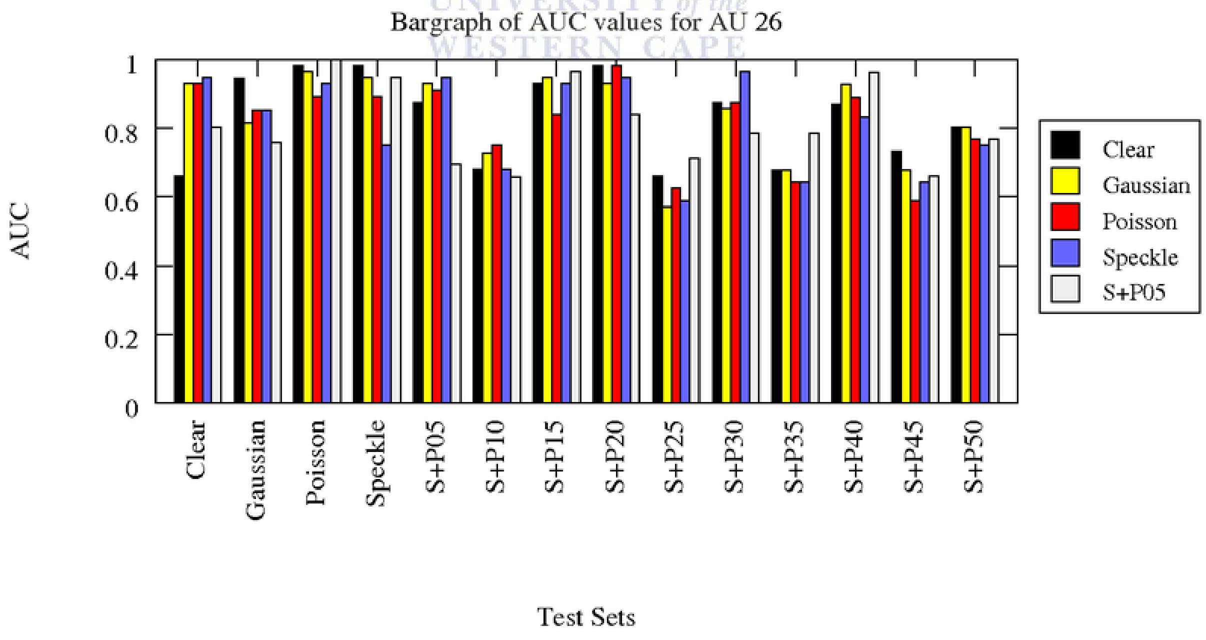


Figure B.16: Bargraph of AUC values for AU 26

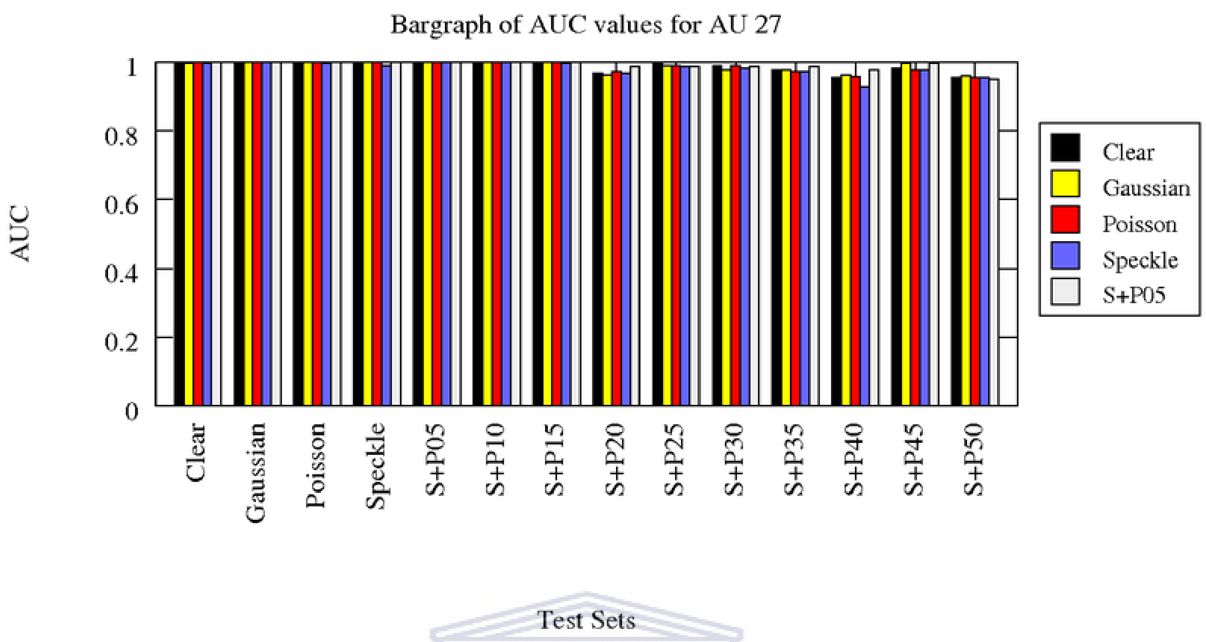


Figure B.17: Bargraph of AUC values for AU 27

Table B.1: AU=1

Training Set	Test Sets														
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	**0.999**	**0.986**	**1.000**	**0.983**	**0.995**	**0.931**	**0.990**	**0.939**	**0.942**	*0.848*	0.711	*0.838*	0.792	0.711	**0.905**
Gaussian	**0.993**	**0.963**	**0.995**	**0.974**	**0.992**	**0.938**	**0.986**	**0.934**	**0.932**	*0.868*	0.733	*0.878*	*0.858*	0.787	**0.916**
Poisson	**0.999**	**0.988**	**0.995**	**0.967**	**0.987**	**0.938**	**0.984**	**0.945**	**0.928**	*0.881*	0.757	*0.893*	*0.827*	0.753	**0.917**
Speckle	**0.988**	**0.980**	**0.988**	**0.944**	**0.983**	**0.945**	**0.984**	**0.926**	**0.923**	*0.899*	*0.826*	*0.892*	*0.864*	*0.803*	**0.925**
S+P05	**0.985**	**0.972**	**0.983**	**0.950**	**0.975**	**0.954**	**0.990**	**0.921**	**0.927**	*0.864*	*0.805*	*0.881*	*0.818*	*0.802*	**0.916**
S+P10	**0.971**	**0.957**	**0.980**	**0.931**	**0.970**	**0.901**	**0.970**	*0.900*	**0.911**	*0.864*	0.791	*0.858*	*0.879*	0.770	**0.904**
S+P15	**0.967**	**0.972**	**0.975**	**0.935**	**0.986**	**0.947**	**0.968**	**0.937**	**0.904**	*0.871*	*0.809*	*0.862*	*0.860*	*0.813*	**0.915**
S+P20	**0.935**	**0.958**	**0.955**	**0.932**	**0.966**	**0.937**	**0.956**	*0.888*	*0.896*	*0.861*	*0.812*	*0.869*	*0.836*	*0.803*	**0.900**
S+P25	**0.948**	**0.942**	**0.943**	**0.919**	**0.948**	**0.933**	**0.960**	*0.870*	*0.896*	*0.892*	*0.801*	*0.861*	*0.856*	*0.825*	*0.900*
S+P30	*0.889*	**0.930**	**0.927**	*0.896*	**0.924**	**0.915**	**0.927**	*0.872*	*0.867*	*0.871*	0.787	*0.850*	*0.804*	*0.813*	*0.877*
S+P35	*0.876*	**0.903**	**0.917**	*0.886*	**0.924**	*0.886*	**0.907**	*0.842*	*0.854*	*0.867*	*0.817*	*0.885*	*0.877*	*0.808*	*0.875*
S+P40	*0.881*	*0.872*	**0.902**	*0.883*	**0.911**	*0.890*	**0.916**	*0.870*	*0.876*	*0.850*	*0.812*	*0.810*	*0.806*	*0.843*	*0.866*
S+P45	*0.887*	**0.903**	**0.921**	*0.896*	**0.934**	*0.894*	**0.912**	*0.864*	*0.850*	*0.885*	0.778	*0.878*	*0.823*	*0.831*	*0.875*
S+P50	*0.848*	*0.886*	**0.903**	*0.883*	*0.896*	*0.868*	*0.893*	*0.856*	*0.855*	*0.873*	0.781	*0.853*	*0.831*	*0.817*	*0.860*
Averages	**0.940**	**0.944**	**0.956**	**0.927**	**0.957**	**0.920**	**0.953**	*0.897*	*0.897*	*0.871*	0.787	*0.865*	*0.838*	0.799	*0.896*

Table B.2: AU=2

Training Set	Test Sets														
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	**0.997**	**1.000**	**0.997**	**0.991**	**0.978**	**0.997**	**0.972**	**0.956**	**0.994**	**0.980**	**0.920**	**0.969**	*0.840*	*0.888*	**0.963**
Gaussian	**0.997**	**1.000**	**0.997**	**0.960**	**0.978**	**0.981**	**0.966**	**0.953**	**0.975**	**0.993**	**0.934**	**0.969**	*0.836*	**0.932**	**0.962**
Poisson	**0.997**	**1.000**	**0.994**	**0.984**	**0.978**	**0.997**	**0.969**	**0.972**	**0.988**	**0.993**	**0.946**	**0.966**	*0.855*	**0.904**	**0.967**
Speckle	**0.994**	**1.000**	**0.994**	**0.938**	**0.997**	**0.981**	**0.975**	**0.966**	**0.951**	**0.980**	**0.924**	**0.957**	*0.840*	*0.895*	**0.957**
S+P05	**0.994**	**1.000**	**0.997**	**0.972**	**0.975**	**0.988**	**0.966**	**0.975**	**0.975**	**0.977**	**0.934**	**0.960**	*0.827*	**0.904**	**0.960**
S+P10	**0.990**	**1.000**	**0.991**	**0.963**	**0.982**	**0.959**	**0.978**	**0.984**	**0.957**	**0.990**	**0.909**	**0.960**	*0.824*	**0.954**	**0.960**
S+P15	**0.990**	**1.000**	**0.988**	**0.957**	**0.972**	**0.984**	**0.947**	**0.981**	**0.963**	**0.984**	**0.913**	**0.938**	*0.852*	**0.935**	**0.957**
S+P20	**0.990**	**1.000**	**0.981**	**0.960**	**0.975**	**0.962**	**0.960**	**0.962**	**0.938**	**0.987**	**0.924**	**0.957**	*0.852*	**0.923**	**0.955**
S+P25	**0.981**	**1.000**	**0.981**	**0.951**	**0.972**	**0.975**	**0.972**	**0.966**	**0.917**	**0.984**	*0.869*	**0.913**	*0.889*	**0.904**	**0.948**
S+P30	**0.975**	**0.993**	**0.981**	**0.957**	**0.975**	**0.950**	**0.957**	**0.978**	**0.929**	**0.971**	*0.865*	**0.944**	*0.833*	**0.910**	**0.944**
S+P35	**0.978**	**1.000**	**0.975**	**0.941**	**0.975**	**0.959**	**0.972**	**0.975**	**0.914**	**0.974**	*0.869*	**0.957**	*0.827*	**0.901**	**0.944**
S+P40	**0.959**	**1.000**	**0.972**	**0.919**	**0.978**	**0.934**	**0.923**	**0.972**	**0.901**	**0.971**	*0.884*	**0.904**	*0.815*	*0.876*	**0.929**
S+P45	**0.984**	**1.000**	**0.978**	**0.935**	**0.969**	**0.953**	**0.929**	**0.947**	**0.954**	**0.938**	*0.895*	**0.963**	*0.830*	**0.944**	**0.944**
S+P50	**0.959**	**1.000**	**0.959**	**0.901**	**0.982**	**0.975**	**0.951**	**0.966**	*0.895*	**0.958**	*0.880*	**0.901**	0.765	*0.864*	**0.925**
Averages	**0.985**	**0.999**	**0.985**	**0.952**	**0.978**	**0.971**	**0.960**	**0.968**	**0.946**	**0.977**	**0.905**	**0.947**	*0.835*	**0.910**	**0.951**

Table B.3: AU=4

Training Set	Test Sets														
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	**0.999**	**0.983**	**0.997**	**0.976**	**0.962**	**0.929**	**0.913**	*0.881*	**0.948**	**0.907**	*0.825*	*0.889*	0.785	0.692	**0.906**
Gaussian	**0.996**	**0.956**	**0.992**	**0.984**	**0.965**	**0.906**	**0.918**	*0.878*	**0.918**	**0.905**	*0.824*	*0.868*	0.763	0.691	*0.897*
Poisson	**1.000**	**0.990**	**0.984**	**0.999**	**0.968**	**0.938**	**0.924**	*0.896*	**0.942**	**0.909**	*0.815*	*0.886*	0.772	0.695	**0.908**
Speckle	**0.992**	**0.957**	**0.993**	**0.978**	**0.953**	*0.881*	*0.878*	**0.913**	**0.912**	**0.936**	*0.817*	*0.862*	0.778	0.708	*0.897*
S+P05	**0.992**	**0.977**	**0.975**	**0.993**	**0.930**	**0.913**	**0.903**	*0.883*	**0.913**	**0.934**	*0.810*	*0.847*	0.760	0.690	*0.894*
S+P10	**0.976**	**0.965**	**0.971**	**0.984**	**0.946**	*0.841*	*0.899*	*0.891*	**0.907**	**0.935**	*0.831*	0.773	0.758	0.747	*0.887*
S+P15	**0.965**	**0.963**	**0.961**	**0.978**	**0.939**	*0.851*	*0.825*	*0.878*	*0.895*	**0.928**	*0.825*	*0.822*	*0.804*	0.698	*0.881*
S+P20	**0.966**	**0.980**	**0.951**	**0.976**	**0.943**	*0.863*	*0.887*	*0.815*	*0.888*	**0.914**	0.781	*0.873*	0.752	0.740	*0.881*
S+P25	**0.946**	**0.949**	**0.938**	**0.966**	**0.920**	*0.852*	*0.841*	*0.816*	*0.828*	*0.881*	0.788	*0.805*	0.775	0.683	*0.856*
S+P30	**0.924**	**0.927**	**0.925**	**0.963**	**0.922**	*0.801*	*0.885*	*0.853*	*0.844*	*0.819*	0.791	0.784	0.792	0.678	*0.851*
S+P35	**0.936**	**0.942**	*0.883*	**0.965**	**0.930**	*0.803*	*0.832*	*0.863*	*0.831*	*0.857*	0.777	*0.851*	0.664	0.686	*0.844*
S+P40	**0.948**	**0.924**	**0.901**	**0.971**	**0.956**	*0.836*	*0.877*	*0.846*	*0.839*	*0.803*	0.791	*0.818*	0.681	0.650	*0.846*
S+P45	**0.906**	*0.895*	*0.833*	**0.949**	**0.928**	0.789	0.794	0.770	0.783	*0.848*	0.754	0.740	0.727	0.641	*0.811*
S+P50	*0.877*	*0.829*	**0.913**	**0.947**	*0.857*	0.791	0.780	0.766	0.782	0.790	0.784	0.789	0.668	0.489	0.790
Averages	**0.959**	**0.946**	**0.944**	**0.973**	**0.937**	*0.857*	*0.868*	*0.854*	*0.874*	*0.883*	*0.801*	*0.829*	0.748	0.678	*0.868*

Table B.4: AU=5

Training Set	Test Sets														
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	**1.000**	**0.996**	**1.000**	**0.992**	**1.000**	**1.000**	**0.970**	**0.938**	**0.967**	*0.895*	**0.949**	0.788	*0.899*	*0.883*	**0.948**
Gaussian	**1.000**	**0.988**	**1.000**	**1.000**	**1.000**	**1.000**	**0.970**	*0.891*	**0.950**	*0.882*	**0.923**	*0.821*	**0.916**	*0.892*	**0.945**
Poisson	**1.000**	**1.000**	**0.988**	**1.000**	**1.000**	**1.000**	**0.966**	**0.943**	**0.962**	*0.899*	**0.944**	0.788	**0.903**	*0.867*	**0.947**
Speckle	**1.000**	**0.996**	**0.996**	**0.958**	**0.983**	**1.000**	**0.961**	*0.876*	**0.933**	*0.870*	**0.940**	0.796	*0.895*	*0.875*	**0.934**
S+P05	**1.000**	**0.992**	**0.996**	**0.979**	**0.987**	**0.996**	**0.983**	**0.943**	**0.913**	*0.899*	**0.944**	*0.850*	*0.887*	**0.917**	**0.949**
S+P10	**1.000**	**0.962**	**0.979**	**0.954**	**0.987**	**1.000**	**0.961**	**0.910**	**0.929**	*0.853*	**0.932**	*0.829*	*0.895*	*0.871*	**0.933**
S+P15	**1.000**	**0.967**	**0.971**	**0.971**	**0.983**	**0.996**	**0.940**	*0.881*	**0.904**	*0.853*	**0.914**	*0.825*	*0.861*	*0.850*	**0.923**
S+P20	**1.000**	**0.950**	**0.946**	**0.958**	**0.983**	**1.000**	**0.953**	*0.871*	**0.908**	*0.857*	**0.919**	*0.833*	*0.882*	*0.887*	**0.925**
S+P25	**1.000**	**0.954**	**0.971**	**0.979**	**0.966**	**0.992**	**0.949**	*0.900*	*0.863*	0.794	**0.932**	0.783	*0.887*	*0.892*	**0.919**
S+P30	**0.992**	*0.883*	**0.942**	**0.942**	**0.971**	**0.958**	**0.940**	*0.886*	**0.929**	0.794	**0.927**	*0.812*	0.798	*0.858*	**0.902**
S+P35	**0.996**	*0.896*	**0.908**	**0.954**	**0.912**	**0.962**	**0.953**	*0.900*	*0.858*	*0.819*	**0.923**	*0.821*	*0.895*	*0.900*	**0.907**
S+P40	**0.996**	*0.871*	**0.925**	*0.896*	**0.962**	**0.958**	**0.944**	*0.881*	*0.883*	*0.840*	*0.889*	0.738	0.781	*0.854*	*0.887*
S+P45	**1.000**	**0.908**	**0.917**	**0.933**	**0.975**	**0.996**	**0.919**	*0.852*	*0.896*	*0.803*	**0.940**	0.783	*0.849*	*0.854*	**0.902**
S+P50	**0.983**	*0.850*	*0.896*	**0.933**	**0.916**	**0.950**	**0.936**	*0.819*	*0.854*	*0.828*	**0.944**	0.729	*0.865*	*0.854*	*0.883*
Averages	**0.998**	**0.944**	**0.960**	**0.961**	**0.973**	**0.986**	0.953	*0.892*	**0.911**	*0.849*	**0.930**	0.800	*0.872*	*0.875*	**0.922**

Table B.5: AU=6

Training Set	Test Sets														
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.997**	**0.957**	**1.000**	**0.995**	**0.904**	*0.877*	*0.876*	0.717	**0.952**
Gaussian	**1.000**	**0.985**	**1.000**	**0.998**	**0.990**	**0.995**	**0.987**	**0.947**	**0.993**	**1.000**	**0.938**	*0.900*	*0.884*	0.725	**0.953**
Poisson	**1.000**	**0.992**	**1.000**	**1.000**	**1.000**	**1.000**	**0.992**	**0.942**	**1.000**	**1.000**	**0.927**	*0.895*	*0.894*	0.733	**0.955**
Speckle	**1.000**	**0.987**	**1.000**	**0.995**	**0.990**	**1.000**	**0.995**	**0.934**	**0.988**	**1.000**	**0.969**	*0.897*	**0.904**	0.725	**0.956**
S+P05	**1.000**	**0.977**	**0.995**	**0.998**	**0.990**	**0.997**	**0.977**	**0.929**	**0.985**	**1.000**	**0.951**	*0.890*	*0.894*	0.744	**0.952**
S+P10	**1.000**	**0.977**	**0.993**	**0.998**	**0.990**	**0.974**	**0.997**	**0.909**	**1.000**	**1.000**	**0.961**	**0.925**	*0.874*	0.731	**0.952**
S+P15	**1.000**	**0.977**	**0.977**	**0.995**	**0.974**	**1.000**	**0.974**	**0.914**	**0.980**	**1.000**	**0.969**	*0.890*	*0.876*	*0.803*	**0.952**
S+P20	**0.998**	**0.982**	**0.980**	**0.995**	**0.979**	**0.969**	**0.985**	*0.876*	**0.980**	**0.987**	**0.951**	*0.897*	*0.889*	0.771	**0.946**
S+P25	**0.995**	**0.985**	**0.977**	**0.998**	**0.995**	**0.997**	**0.990**	**0.907**	**0.970**	**0.964**	**0.938**	**0.925**	**0.917**	0.768	**0.952**
S+P30	**0.998**	**0.985**	**0.980**	**0.992**	**0.992**	**0.987**	**0.987**	*0.891*	**0.975**	**0.959**	**0.930**	*0.897*	*0.876*	0.781	**0.945**
S+P35	**0.970**	**0.949**	**0.975**	**0.980**	**0.977**	**0.941**	**0.967**	*0.861*	**0.983**	**0.967**	**0.945**	*0.900*	*0.889*	0.773	**0.934**
S+P40	**0.987**	**0.964**	**0.965**	**0.990**	**0.971**	**0.946**	**0.992**	**0.914**	**0.975**	**0.944**	**0.958**	*0.808*	*0.896*	0.728	**0.931**
S+P45	**0.995**	**0.921**	**0.927**	**0.970**	**0.919**	**0.936**	**0.936**	**0.912**	**0.972**	**0.962**	**0.935**	0.770	*0.851*	0.712	**0.908**
S+P50	**0.987**	**0.928**	**0.930**	**0.967**	**0.924**	**0.913**	**0.977**	*0.881*	**0.952**	**0.903**	*0.898*	*0.860*	*0.846*	0.600	*0.898*
Averages	**0.995**	**0.972**	**0.979**	**0.991**	**0.978**	**0.975**	**0.982**	**0.912**	**0.982**	**0.977**	**0.941**	*0.881*	*0.883*	0.736	**0.942**

Table B.6: AU=7

Training Set	Test Sets														
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	**0.979**	**0.990**	**1.000**	**0.983**	**0.964**	**0.978**	**0.953**	**0.947**	**0.943**	*0.864*	*0.854*	0.781	*0.848*	0.789	**0.919**
Gaussian	**0.979**	**0.952**	**1.000**	**0.988**	**0.971**	**0.973**	**0.973**	**0.937**	**0.924**	*0.857*	*0.873*	*0.829*	*0.804*	0.792	**0.918**
Poisson	**0.981**	**0.995**	**1.000**	**0.985**	**0.974**	**0.966**	**0.956**	**0.937**	**0.938**	*0.864*	*0.868*	*0.805*	*0.846*	0.775	**0.921**
Speckle	**0.979**	**0.998**	**1.000**	**0.941**	**0.976**	**0.985**	**0.961**	**0.903**	**0.931**	*0.873*	*0.890*	0.795	*0.858*	0.787	**0.920**
S+P05	**0.960**	**0.995**	**1.000**	**0.985**	**0.940**	**0.976**	**0.941**	**0.923**	**0.905**	*0.864*	*0.857*	*0.821*	0.797	0.797	**0.912**
S+P10	**0.974**	**0.995**	**1.000**	**0.958**	**0.971**	**0.935**	**0.936**	**0.932**	**0.902**	*0.849*	*0.849*	*0.826*	*0.848*	0.749	**0.909**
S+P15	**0.910**	**0.973**	**0.993**	**0.978**	**0.969**	**0.959**	*0.858*	*0.882*	**0.907**	*0.859*	*0.849*	0.790	*0.843*	0.770	*0.896*
S+P20	**0.967**	**0.966**	**1.000**	**0.939**	**0.959**	**0.942**	**0.951**	0.787	*0.840*	*0.804*	*0.885*	*0.802*	*0.816*	0.758	*0.887*
S+P25	**0.931**	**0.971**	**0.995**	**0.968**	**0.940**	*0.896*	*0.897*	*0.831*	*0.855*	0.787	*0.854*	*0.824*	*0.831*	0.720	*0.879*
S+P30	**0.912**	**0.906**	**0.988**	**0.968**	**0.916**	*0.894*	*0.890*	0.763	*0.876*	0.770	*0.854*	0.743	*0.843*	0.766	*0.863*
S+P35	**0.948**	**0.928**	**1.000**	**0.939**	**0.952**	**0.935**	**0.904**	*0.836*	*0.859*	*0.825*	*0.849*	0.731	0.772	0.751	*0.873*
S+P40	**0.905**	**0.901**	**1.000**	**0.919**	**0.933**	**0.915**	**0.902**	*0.860*	*0.862*	*0.842*	*0.852*	0.719	*0.806*	*0.816*	*0.874*
S+P45	*0.876*	*0.874*	**0.980**	*0.882*	*0.888*	**0.928**	*0.850*	*0.819*	*0.831*	0.770	*0.818*	0.688	0.735	0.734	*0.834*
S+P50	*0.836*	*0.877*	*0.873*	*0.885*	*0.878*	*0.865*	*0.838*	0.734	*0.867*	0.780	*0.813*	0.688	0.784	0.701	*0.816*
Averages	**0.938**	**0.951**	**0.988**	**0.951**	**0.945**	**0.939**	**0.915**	*0.864*	*0.889*	*0.829*	*0.855*	0.774	*0.817*	0.765	*0.887*

Table B.7: AU=9

Training Set	Test Sets														
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.978**	**1.000**	*0.802*	**0.989**	*0.844*	*0.808*	**0.959**
Gaussian	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.989**	**0.980**	**0.980**	**0.967**	**0.990**	*0.879*	**0.978**	*0.875*	0.788	**0.959**
Poisson	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.980**	**0.989**	**1.000**	*0.868*	**1.000**	*0.885*	0.778	**0.964**
Speckle	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.980**	**0.956**	**0.960**	*0.890*	**0.945**	*0.896*	*0.879*	**0.965**
S+P05	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.980**	**0.989**	**0.990**	*0.857*	**0.989**	*0.896*	0.788	**0.964**
S+P10	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.956**	**0.980**	**0.940**	**1.000**	**0.980**	*0.890*	**0.978**	*0.844*	*0.859*	**0.959**
S+P15	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.934**	**0.980**	**0.940**	**0.978**	**0.980**	*0.879*	**0.934**	*0.896*	*0.859*	**0.956**
S+P20	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.912**	**1.000**	*0.880*	**0.923**	**0.919**	*0.846*	**0.978**	*0.885*	0.788	**0.938**
S+P25	**1.000**	**1.000**	**0.958**	**1.000**	**0.990**	**1.000**	**0.960**	**0.910**	**0.912**	**0.939**	**0.967**	**0.978**	**0.906**	*0.828*	**0.953**
S+P30	**0.990**	**1.000**	**1.000**	**1.000**	**0.990**	**0.934**	**1.000**	*0.900*	**0.912**	**1.000**	*0.824*	**0.989**	0.781	*0.828*	**0.939**
S+P35	**0.960**	**1.000**	**0.990**	**1.000**	**0.948**	**0.923**	**1.000**	*0.870*	**0.956**	**0.950**	*0.824*	**0.934**	*0.823*	0.758	**0.924**
S+P40	**1.000**	**1.000**	**0.948**	**0.967**	**0.990**	**0.945**	**1.000**	**0.920**	**0.967**	**0.939**	*0.890*	**0.934**	*0.823*	*0.859*	**0.942**
S+P45	**0.990**	**1.000**	**0.927**	**1.000**	**1.000**	**0.945**	*0.889*	*0.890*	**0.923**	**0.939**	0.758	**0.923**	0.781	*0.879*	**0.917**
S+P50	**0.930**	**1.000**	**0.917**	**1.000**	**0.979**	**0.901**	**0.990**	0.750	*0.890*	**0.929**	0.714	*0.879*	0.792	0.737	*0.886*
Averages	**0.991**	**1.000**	**0.981**	**0.998**	**0.993**	**0.960**	**0.984**	**0.923**	**0.953**	**0.965**	*0.849*	**0.959**	*0.852*	*0.817*	**0.945**

Table B.8: AU=11

Training Set	Test Sets														
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	**1.000**	**1.000**	**1.000**	**0.943**	**1.000**	**0.972**	**1.000**	**0.944**	**1.000**	**0.944**	**1.000**	*0.806*	**0.971**	*0.812*	**0.957**
Gaussian	**1.000**	*0.857*	**1.000**	**0.971**	**1.000**	*0.861*	**1.000**	**1.000**	**0.943**	**1.000**	**1.000**	0.694	**0.943**	*0.875*	**0.939**
Poisson	**1.000**	**1.000**	**0.971**	**0.943**	**1.000**	**1.000**	**1.000**	**0.944**	**0.943**	*0.889*	**1.000**	0.750	**1.000**	0.719	**0.940**
Speckle	**1.000**	**0.914**	**1.000**	0.743	**1.000**	*0.889*	**1.000**	**0.944**	**1.000**	**0.917**	**1.000**	0.722	**1.000**	0.781	**0.922**
S+P05	**1.000**	*0.886*	**1.000**	*0.829*	**0.971**	*0.861*	**1.000**	**0.972**	**0.971**	**0.917**	**1.000**	0.722	**0.943**	*0.875*	**0.925**
S+P10	**1.000**	*0.886*	**1.000**	**1.000**	**1.000**	*0.889*	**1.000**	**0.972**	**1.000**	**1.000**	**1.000**	0.750	**0.971**	*0.875*	**0.953**
S+P15	**1.000**	**1.000**	**1.000**	**0.943**	**1.000**	**0.972**	**1.000**	**0.944**	**1.000**	0.778	**1.000**	0.778	**0.971**	0.688	**0.934**
S+P20	*0.889*	**0.943**	**0.971**	*0.886*	**1.000**	**0.972**	**1.000**	*0.861*	**0.914**	*0.806*	**1.000**	0.694	*0.829*	*0.844*	**0.901**
S+P25	**0.917**	**0.971**	**0.971**	**1.000**	**1.000**	**0.944**	**0.971**	*0.806*	*0.886*	*0.833*	**1.000**	0.694	**0.943**	*0.812*	**0.911**
S+P30	*0.889*	**0.971**	**0.943**	**0.914**	**1.000**	**0.972**	**0.943**	*0.889*	**0.943**	0.583	**1.000**	*0.833*	**0.943**	0.531	*0.882*
S+P35	**0.917**	**1.000**	**1.000**	**1.000**	**1.000**	**0.917**	**0.971**	*0.889*	0.771	*0.833*	0.636	0.583	*0.857*	0.750	*0.866*
S+P40	**0.944**	*0.829*	**0.971**	*0.857*	**0.943**	*0.861*	*0.886*	**0.917**	*0.829*	**0.944**	**1.000**	0.694	*0.829*	*0.844*	*0.882*
S+P45	*0.833*	**1.000**	**0.914**	*0.886*	*0.800*	**0.944**	**1.000**	0.722	*0.886*	0.722	**0.909**	0.694	0.457	*0.844*	*0.829*
S+P50	0.500	0.629	**0.943**	**0.914**	*0.886*	*0.806*	**0.914**	*0.861*	*0.886*	0.639	*0.818*	**0.972**	**1.000**	0.750	*0.823*
Averages	**0.921**	**0.920**	**0.977**	**0.916**	**0.971**	**0.919**	**0.977**	**0.905**	**0.927**	*0.843*	**0.955**	0.742	**0.904**	0.786	**0.904**

Table B.9: AU=12

Training Set	Test Sets														
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	**0.994**	**1.000**	**1.000**	**0.996**	**0.996**	**0.985**	**0.989**	**0.992**	**1.000**	**0.912**	**0.977**	*0.882*	*0.865*	0.748	**0.953**
Gaussian	**0.994**	**0.998**	**1.000**	**0.989**	**0.992**	**0.976**	**0.985**	**0.989**	**0.998**	**0.938**	**0.977**	*0.886*	*0.861*	0.740	**0.952**
Poisson	**0.998**	**1.000**	**0.996**	**0.996**	**0.996**	**0.992**	**0.985**	**0.996**	**1.000**	**0.906**	**0.969**	*0.879*	*0.880*	0.748	**0.953**
Speckle	**0.990**	**1.000**	**1.000**	**0.977**	**0.992**	**0.976**	**0.985**	**0.968**	**1.000**	**0.925**	**0.963**	*0.855*	*0.880*	0.750	**0.947**
S+P05	**0.996**	**1.000**	**1.000**	**0.989**	**0.985**	**0.972**	**0.970**	**0.979**	**0.998**	**0.932**	**0.950**	*0.875*	*0.865*	0.760	**0.948**
S+P10	**0.967**	**0.998**	**0.996**	**0.981**	**0.985**	**0.955**	**0.959**	**0.976**	**0.996**	*0.897*	**0.948**	*0.868*	*0.884*	0.788	**0.943**
S+P15	**0.979**	**1.000**	**0.994**	**0.973**	**0.985**	**0.949**	**0.950**	**0.981**	**0.996**	**0.932**	**0.930**	*0.848*	**0.907**	0.783	**0.943**
S+P20	**0.973**	**0.992**	**0.992**	**0.962**	**0.987**	**0.949**	**0.941**	**0.947**	**0.994**	**0.908**	**0.936**	*0.862*	*0.878*	0.735	**0.933**
S+P25	**0.973**	**0.992**	**0.994**	**0.964**	**0.966**	**0.942**	**0.959**	**0.957**	**0.990**	*0.895*	**0.942**	*0.877*	**0.901**	0.775	**0.938**
S+P30	**0.967**	**0.994**	**0.994**	**0.966**	**0.987**	**0.947**	**0.950**	**0.979**	**0.990**	**0.908**	**0.946**	*0.855*	**0.907**	0.779	**0.941**
S+P35	**0.961**	**0.981**	**0.990**	**0.943**	**0.958**	**0.929**	*0.900*	**0.942**	**0.969**	*0.882*	*0.890*	*0.859*	*0.876*	0.744	**0.916**
S+P40	**0.930**	**0.967**	**0.994**	**0.956**	**0.970**	**0.912**	**0.939**	**0.959**	**0.992**	*0.868*	**0.928**	*0.817*	*0.840*	0.769	**0.917**
S+P45	**0.938**	**0.988**	**0.985**	**0.920**	**0.970**	**0.927**	**0.935**	**0.938**	**0.975**	**0.904**	**0.955**	*0.841*	*0.897*	0.752	**0.923**
S+P50	**0.952**	**0.971**	**0.967**	*0.899*	**0.954**	*0.876*	**0.913**	*0.885*	**0.963**	*0.870*	*0.880*	*0.853*	**0.914**	0.719	**0.901**
Averages	**0.972**	**0.992**	**0.993**	**0.965**	**0.980**	**0.949**	**0.954**	**0.963**	**0.990**	**0.905**	**0.942**	*0.861*	*0.882*	0.756	**0.936**

Table B.10: AU=15

Training Set	Test Sets														
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	**0.957**	**0.976**	**0.974**	**0.914**	**0.962**	**0.934**	*0.843*	*0.884*	*0.828*	*0.881*	0.774	0.724	0.764	0.716	*0.866*
Gaussian	**0.957**	**0.947**	**0.995**	**0.914**	**0.952**	**0.944**	*0.857*	**0.919**	*0.814*	*0.895*	0.789	0.757	0.731	0.702	*0.869*
Poisson	**0.976**	**0.990**	**0.937**	*0.871*	**0.948**	**0.929**	*0.800*	*0.859*	0.779	*0.852*	0.779	0.705	0.736	0.707	*0.848*
Speckle	**0.943**	**0.971**	**0.968**	0.771	**0.924**	*0.889*	*0.852*	*0.899*	0.775	**0.910**	0.789	0.700	0.692	0.639	*0.837*
S+P05	**0.976**	**0.966**	**0.958**	*0.852*	*0.857*	**0.914**	0.781	*0.879*	0.716	*0.881*	*0.822*	0.733	0.745	0.707	*0.842*
S+P10	**0.952**	**0.971**	**0.984**	*0.838*	*0.900*	*0.823*	*0.814*	**0.919**	0.735	*0.833*	0.755	0.686	0.644	0.673	*0.823*
S+P15	*0.881*	**0.961**	**0.942**	*0.857*	*0.867*	*0.879*	0.648	*0.874*	0.721	*0.876*	0.731	0.638	0.664	0.630	0.798
S+P20	*0.891*	**0.961**	**0.926**	*0.871*	**0.924**	*0.874*	0.762	*0.808*	*0.853*	*0.881*	*0.832*	0.729	0.702	0.630	*0.832*
S+P25	**0.929**	**0.952**	**0.953**	*0.824*	*0.871*	**0.919**	0.738	*0.899*	0.701	*0.800*	*0.817*	0.714	0.688	0.630	*0.817*
S+P30	**0.933**	**0.914**	**0.947**	0.762	*0.867*	0.798	*0.838*	0.798	*0.809*	0.733	0.731	0.619	0.707	0.644	0.793
S+P35	*0.867*	**0.933**	**0.926**	0.781	*0.862*	*0.869*	*0.805*	*0.869*	0.750	*0.857*	0.731	0.686	0.673	0.606	*0.801*
S+P40	*0.824*	**0.923**	*0.816*	0.771	*0.824*	*0.838*	0.748	*0.843*	0.765	0.795	*0.832*	0.467	0.649	0.635	0.766
S+P45	*0.867*	*0.875*	*0.837*	0.719	*0.829*	0.773	0.695	*0.803*	0.770	0.695	0.760	0.633	0.606	0.567	0.745
S+P50	0.652	*0.808*	*0.853*	0.724	*0.886*	*0.838*	0.733	*0.854*	0.711	*0.848*	0.611	0.552	0.678	0.490	0.731
Averages	**0.900**	**0.939**	**0.930**	*0.819*	*0.891*	*0.873*	0.780	*0.865*	0.766	*0.838*	0.768	0.667	0.691	0.641	*0.812*

Table B.11: AU=17

Training Set	Test Sets														
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	**0.946**	**0.902**	**0.958**	**0.921**	**0.930**	*0.887*	**0.938**	*0.893*	**0.903**	*0.891*	*0.851*	*0.862*	0.688	0.790	*0.883*
Gaussian	**0.957**	*0.844*	**0.915**	**0.934**	**0.934**	*0.887*	**0.909**	**0.907**	*0.886*	*0.880*	*0.849*	*0.891*	0.734	*0.804*	*0.881*
Poisson	**0.966**	**0.910**	**0.903**	**0.922**	**0.923**	*0.899*	**0.923**	*0.898*	*0.889*	*0.893*	*0.838*	*0.869*	0.711	0.791	*0.881*
Speckle	**0.939**	*0.895*	*0.900*	*0.896*	**0.917**	*0.883*	**0.930**	**0.903**	*0.862*	*0.887*	*0.845*	*0.854*	0.732	*0.812*	*0.875*
S+P05	**0.927**	**0.910**	**0.935**	**0.913**	*0.858*	*0.879*	**0.928**	*0.899*	*0.886*	*0.868*	*0.853*	*0.868*	0.705	*0.825*	*0.875*
S+P10	**0.908**	*0.869*	**0.914**	**0.911**	*0.896*	*0.833*	*0.878*	**0.904**	*0.882*	*0.853*	*0.879*	*0.837*	0.766	*0.816*	*0.868*
S+P15	*0.896*	*0.869*	*0.861*	*0.879*	**0.908**	*0.878*	0.793	*0.900*	*0.839*	*0.849*	*0.810*	*0.830*	0.718	0.732	*0.840*
S+P20	*0.894*	*0.855*	*0.873*	*0.857*	*0.880*	*0.865*	*0.868*	*0.816*	*0.844*	*0.876*	*0.834*	*0.846*	0.766	*0.812*	*0.849*
S+P25	*0.874*	*0.873*	*0.844*	*0.851*	*0.897*	*0.863*	*0.864*	*0.867*	0.766	*0.861*	*0.844*	*0.830*	0.717	0.778	*0.838*
S+P30	*0.876*	*0.830*	*0.856*	*0.863*	*0.855*	*0.827*	*0.860*	**0.912**	*0.868*	*0.828*	*0.842*	*0.835*	0.693	*0.815*	*0.840*
S+P35	*0.842*	*0.830*	*0.806*	*0.851*	*0.859*	*0.844*	*0.815*	*0.874*	0.780	0.778	0.763	*0.835*	0.748	0.732	*0.811*
S+P40	*0.881*	*0.840*	*0.842*	*0.825*	*0.889*	*0.859*	*0.838*	**0.909**	0.728	*0.869*	*0.831*	*0.817*	0.723	*0.805*	*0.833*
S+P45	*0.817*	*0.822*	0.781	0.749	*0.825*	*0.813*	*0.833*	*0.822*	0.744	*0.892*	0.785	*0.833*	0.697	0.786	0.800
S+P50	*0.814*	0.783	0.786	0.791	*0.803*	0.796	0.658	*0.825*	0.744	*0.813*	0.763	*0.826*	0.656	0.680	0.767
Averages	*0.896*	*0.859*	*0.870*	*0.869*	*0.884*	*0.858*	*0.860*	*0.881*	*0.830*	*0.860*	*0.828*	*0.845*	0.718	0.784	*0.846*

Table B.12: AU=20

Training Set	Test Sets														
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	**0.988**	**1.000**	**1.000**	**1.000**	**0.988**	**0.970**	**0.982**	**0.988**	**0.929**	**0.954**	*0.805*	**0.910**	0.762	*0.851*	**0.938**
Gaussian	**0.982**	**1.000**	**1.000**	**0.994**	**0.970**	**0.951**	**0.970**	**0.994**	**0.923**	**0.948**	*0.840*	**0.932**	0.768	*0.845*	**0.937**
Poisson	**0.994**	**1.000**	**1.000**	**1.000**	**1.000**	**0.970**	**0.964**	**0.988**	**0.923**	**0.935**	*0.840*	**0.917**	0.738	*0.833*	**0.936**
Speckle	**0.988**	**0.990**	**1.000**	**0.976**	**0.982**	**0.939**	**0.927**	**0.988**	**0.905**	**0.948**	*0.858*	**0.925**	0.774	*0.833*	**0.931**
S+P05	**0.994**	**1.000**	**0.994**	**1.000**	*0.851*	**0.915**	**0.939**	**0.988**	*0.857*	**0.922**	*0.852*	**0.925**	0.774	*0.839*	**0.918**
S+P10	**0.982**	**1.000**	**0.988**	**1.000**	**0.934**	*0.897*	**0.927**	**0.982**	*0.881*	*0.856*	*0.893*	*0.887*	0.732	*0.821*	**0.913**
S+P15	**0.959**	**1.000**	**0.981**	**0.994**	**0.982**	*0.891*	*0.842*	**0.982**	*0.887*	**0.915**	*0.822*	**0.925**	0.691	*0.804*	**0.905**
S+P20	**0.941**	**1.000**	**0.981**	**1.000**	**0.934**	**0.946**	*0.885*	**0.964**	*0.869*	*0.823*	*0.834*	**0.902**	0.714	*0.857*	**0.904**
S+P25	**0.935**	**0.971**	**0.975**	**0.988**	*0.893*	**0.939**	**0.909**	**0.988**	0.780	*0.895*	*0.852*	*0.895*	0.756	*0.809*	*0.899*
S+P30	**0.941**	**1.000**	**1.000**	**1.000**	**0.964**	**0.964**	*0.830*	**0.970**	*0.893*	*0.804*	0.799	*0.872*	0.684	*0.851*	*0.898*
S+P35	*0.882*	**0.914**	**0.975**	**0.976**	*0.839*	*0.854*	**0.903**	**0.951**	*0.851*	*0.830*	0.793	**0.902**	0.780	*0.804*	*0.875*
S+P40	**0.929**	**1.000**	**0.988**	**0.994**	*0.887*	**0.939**	*0.836*	**0.958**	0.798	*0.882*	0.746	*0.887*	0.720	*0.881*	*0.889*
S+P45	**0.929**	*0.867*	**0.962**	**0.929**	*0.887*	**0.909**	*0.891*	**0.976**	*0.821*	0.791	*0.828*	**0.940**	0.613	0.708	*0.861*
S+P50	*0.834*	**0.943**	**0.925**	**1.000**	*0.821*	**0.903**	0.727	*0.867*	*0.845*	0.765	0.716	*0.835*	*0.839*	*0.899*	*0.851*
Averages	**0.948**	**0.977**	**0.984**	**0.989**	**0.924**	**0.928**	*0.895*	**0.970**	*0.869*	*0.876*	*0.820*	**0.904**	0.739	*0.831*	**0.904**

Table B.13: AU=23

Training Set	Test Sets														
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	*0.870*	**0.955**	**1.000**	**1.000**	**1.000**	*0.900*	**1.000**	*0.820*	**1.000**	*0.806*	**0.940**	*0.857*	0.741	0.661	*0.896*
Gaussian	**1.000**	*0.886*	**1.000**	**1.000**	**0.980**	**0.940**	*0.885*	*0.860*	**1.000**	*0.806*	**0.960**	*0.857*	0.722	0.732	**0.902**
Poisson	**1.000**	**0.977**	**1.000**	**1.000**	**0.980**	**0.920**	**1.000**	**0.920**	**1.000**	*0.861*	**0.960**	*0.804*	0.611	0.696	**0.909**
Speckle	**0.963**	**0.955**	**1.000**	**0.926**	**0.940**	*0.900*	**1.000**	0.780	**1.000**	0.750	**1.000**	*0.821*	0.667	0.732	*0.888*
S+P05	**0.963**	**0.955**	**1.000**	**0.982**	*0.900*	**0.960**	**0.961**	*0.820*	**1.000**	*0.861*	**0.940**	0.786	0.667	0.714	*0.893*
S+P10	**0.963**	**0.977**	**1.000**	**0.963**	**1.000**	*0.880*	**0.961**	**0.980**	**1.000**	*0.861*	**1.000**	0.768	0.685	0.732	**0.912**
S+P15	**0.963**	**0.932**	**1.000**	**0.982**	**0.940**	**0.920**	0.615	**0.940**	**1.000**	0.694	**0.940**	*0.893*	0.704	*0.804*	*0.880*
S+P20	**0.963**	*0.886*	**1.000**	**0.926**	*0.880*	*0.860*	**0.961**	*0.800*	**1.000**	*0.806*	**0.920**	0.768	0.704	0.732	*0.872*
S+P25	**0.926**	**0.955**	**1.000**	**1.000**	**1.000**	**0.960**	**0.961**	0.740	**1.000**	**0.917**	**1.000**	**0.964**	*0.833*	0.768	**0.930**
S+P30	**0.944**	**0.977**	**0.955**	**0.982**	**1.000**	*0.840*	**1.000**	*0.860*	**1.000**	0.778	**0.940**	0.696	0.778	*0.804*	*0.897*
S+P35	*0.870*	*0.841*	**1.000**	**0.982**	**0.940**	*0.860*	**1.000**	*0.820*	**1.000**	*0.833*	*0.880*	0.571	*0.833*	*0.839*	*0.876*
S+P40	*0.852*	*0.886*	0.773	**0.926**	**0.920**	*0.880*	**0.961**	0.780	**1.000**	0.778	**0.940**	0.661	0.611	0.679	*0.832*
S+P45	*0.852*	0.773	*0.818*	**0.907**	**0.940**	**0.960**	**0.923**	0.760	**1.000**	0.667	*0.860*	0.661	0.667	0.750	*0.824*
S+P50	**0.982**	*0.864*	*0.818*	*0.833*	**0.960**	*0.860*	*0.846*	*0.900*	**1.000**	0.750	**0.920**	0.589	0.685	0.607	*0.830*
Averages	**0.937**	**0.916**	**0.955**	**0.958**	**0.956**	**0.903**	**0.934**	*0.841*	**1.000**	0.798	**0.943**	0.764	0.708	0.732	*0.882*

Table B.14: AU=24

Training Set	Test Sets														
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	**0.914**	**0.917**	**1.000**	**1.000**	**0.985**	*0.833*	*0.886*	**0.944**	*0.900*	0.769	*0.867*	*0.861*	**0.903**	0.712	*0.892*
Gaussian	**1.000**	0.783	**0.957**	**0.986**	**1.000**	0.788	*0.886*	**0.972**	*0.857*	0.673	*0.883*	*0.847*	*0.819*	0.742	*0.871*
Poisson	**0.986**	*0.900*	*0.857*	**1.000**	**1.000**	*0.833*	**0.943**	**0.944**	**0.929**	0.692	*0.850*	*0.819*	*0.875*	0.758	*0.885*
Speckle	**0.943**	*0.900*	**0.971**	0.708	**0.939**	0.727	*0.814*	**0.958**	*0.871*	0.750	*0.900*	**0.903**	*0.875*	0.697	*0.854*
S+P05	**0.914**	**0.933**	**0.986**	**0.944**	**0.955**	0.773	*0.886*	*0.889*	**0.914**	0.731	*0.833*	*0.875*	**0.917**	0.697	*0.875*
S+P10	**0.943**	*0.850*	**0.971**	**0.944**	**1.000**	0.591	*0.814*	*0.819*	0.786	0.769	*0.867*	**0.931**	*0.889*	0.727	*0.850*
S+P15	**0.986**	**0.967**	**0.971**	*0.889*	**0.955**	*0.818*	0.729	0.750	*0.843*	0.750	*0.867*	*0.847*	*0.833*	0.727	*0.852*
S+P20	**0.929**	**0.967**	**0.986**	**1.000**	**1.000**	0.773	*0.871*	0.792	**0.929**	0.769	*0.900*	*0.861*	*0.889*	0.697	*0.883*
S+P25	**0.943**	**0.917**	**0.929**	**0.917**	**0.970**	*0.803*	**0.971**	*0.833*	0.614	0.615	**0.917**	*0.833*	*0.833*	0.742	*0.845*
S+P30	**0.971**	*0.867*	*0.857*	0.764	**0.909**	0.742	**0.929**	0.583	*0.886*	0.404	**1.000**	0.694	*0.847*	*0.879*	*0.809*
S+P35	*0.886*	*0.817*	**0.929**	*0.833*	**0.955**	0.773	*0.843*	*0.819*	0.743	0.539	**0.967**	0.778	*0.861*	0.773	*0.823*
S+P40	*0.886*	*0.867*	0.757	*0.806*	**0.939**	0.712	*0.900*	*0.819*	0.643	0.442	0.750	0.708	*0.847*	0.727	0.772
S+P45	**1.000**	*0.883*	*0.857*	0.708	*0.849*	*0.864*	*0.857*	*0.806*	*0.900*	0.461	*0.900*	*0.819*	0.722	0.667	*0.807*
S+P50	*0.857*	0.767	**0.957**	0.708	**0.924**	0.773	**0.914**	*0.806*	*0.843*	0.442	*0.800*	0.722	0.764	0.606	0.777
Averages	**0.940**	*0.881*	**0.928**	*0.872*	**0.956**	0.772	*0.874*	*0.838*	*0.833*	0.629	*0.879*	*0.821*	*0.848*	0.725	*0.843*

Table B.15: AU=25

Training Set	Test Sets														
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	**0.967**	**0.979**	**0.975**	**0.961**	**0.950**	**0.910**	**0.943**	*0.893*	**0.947**	*0.874*	*0.827*	0.776	0.683	0.762	*0.889*
Gaussian	**0.959**	**0.971**	**0.961**	**0.960**	**0.945**	**0.919**	**0.932**	*0.893*	**0.954**	*0.873*	0.795	0.783	0.701	0.763	*0.886*
Poisson	**0.975**	**0.980**	**0.958**	**0.973**	**0.947**	**0.916**	**0.942**	*0.896*	**0.944**	*0.881*	*0.826*	0.770	0.704	0.773	*0.892*
Speckle	**0.946**	**0.978**	**0.956**	**0.934**	**0.932**	*0.899*	**0.931**	*0.887*	**0.953**	*0.869*	*0.816*	0.786	0.689	0.764	*0.881*
S+P05	**0.949**	**0.973**	**0.950**	**0.949**	**0.916**	*0.894*	**0.933**	*0.882*	**0.941**	*0.882*	*0.817*	0.774	0.686	0.772	*0.880*
S+P10	**0.940**	**0.964**	**0.941**	**0.934**	**0.926**	*0.884*	**0.925**	*0.877*	**0.932**	*0.866*	0.792	0.789	0.725	0.771	*0.876*
S+P15	**0.915**	**0.961**	**0.915**	**0.934**	**0.925**	*0.895*	**0.911**	*0.874*	**0.928**	*0.862*	0.789	0.795	0.741	0.762	*0.872*
S+P20	**0.905**	**0.953**	**0.907**	**0.923**	**0.914**	*0.879*	**0.908**	*0.862*	**0.916**	*0.873*	0.794	*0.811*	0.718	0.778	*0.867*
S+P25	*0.897*	**0.962**	**0.905**	**0.921**	**0.905**	*0.889*	**0.917**	*0.867*	**0.910**	*0.865*	0.791	0.789	0.730	0.750	*0.864*
S+P30	*0.896*	**0.947**	*0.890*	*0.897*	*0.885*	*0.876*	**0.910**	*0.869*	**0.919**	*0.841*	0.777	0.799	0.723	0.773	*0.857*
S+P35	*0.871*	**0.948**	*0.870*	*0.887*	*0.876*	*0.864*	*0.895*	*0.857*	**0.902**	*0.835*	0.785	0.784	0.718	0.760	*0.847*
S+P40	*0.849*	**0.937**	*0.855*	*0.878*	*0.870*	*0.872*	*0.885*	*0.835*	**0.912**	*0.845*	0.758	0.774	0.710	0.747	*0.838*
S+P45	*0.821*	**0.902**	*0.863*	*0.852*	*0.841*	*0.850*	*0.878*	*0.813*	*0.879*	*0.854*	0.747	0.764	0.679	0.735	*0.820*
S+P50	*0.804*	*0.898*	*0.820*	*0.838*	*0.837*	*0.822*	*0.866*	*0.816*	*0.872*	*0.810*	0.765	0.759	0.668	0.715	*0.806*
Averages	**0.907**	**0.954**	**0.912**	**0.917**	**0.905**	*0.884*	**0.913**	*0.866*	**0.922**	*0.859*	0.791	0.782	0.705	0.759	*0.863*

Table B.16: AU=26

Training Set	Test Sets													
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	0.661	**0.944**	**0.982**	**0.982**	*0.875*	0.682	**0.929**	**0.982**	0.661	*0.875*	0.679	*0.870*	0.732	*0.804*	*0.833*
Gaussian	**0.929**	*0.815*	**0.964**	**0.946**	**0.929**	0.727	**0.946**	**0.929**	0.571	*0.857*	0.679	**0.926**	0.679	*0.804*	*0.836*
Poisson	**0.929**	*0.852*	*0.893*	*0.893*	**0.911**	0.750	*0.839*	**0.982**	0.625	*0.875*	0.643	*0.889*	0.589	0.768	*0.817*
Speckle	**0.946**	*0.852*	**0.929**	0.750	**0.946**	0.682	**0.929**	**0.946**	0.589	**0.964**	0.643	*0.833*	0.643	0.750	*0.814*
S+P05	*0.804*	0.759	**1.000**	**0.946**	0.696	0.659	**0.964**	*0.839*	0.714	0.786	0.786	**0.963**	0.661	0.768	*0.810*
S+P10	**0.982**	*0.870*	**0.982**	**0.982**	*0.875*	0.591	**0.929**	*0.875*	0.732	*0.893*	0.661	*0.889*	0.696	0.768	*0.837*
S+P15	**0.929**	0.796	*0.839*	*0.893*	**0.946**	0.682	0.661	**0.911**	0.446	**0.929**	0.661	*0.870*	0.643	*0.804*	0.786
S+P20	**1.000**	*0.852*	**1.000**	**0.911**	0.786	*0.818*	*0.804*	0.554	0.518	*0.893*	0.786	0.667	0.786	0.786	0.797
S+P25	0.714	*0.889*	**1.000**	*0.875*	*0.893*	0.659	0.768	**0.982**	0.518	**0.911**	0.750	*0.889*	0.500	0.750	0.793
S+P30	**0.911**	0.685	*0.821*	**0.982**	*0.857*	0.659	0.786	0.732	0.571	0.643	0.536	0.778	0.714	*0.875*	0.754
S+P35	**0.911**	*0.852*	*0.875*	**0.964**	*0.839*	0.614	0.786	0.768	0.375	**0.982**	0.571	**0.926**	*0.821*	0.679	0.783
S+P40	0.786	*0.815*	0.750	**0.946**	0.714	0.432	0.750	0.714	0.500	*0.804*	*0.839*	0.741	0.679	*0.857*	0.738
S+P45	*0.839*	0.704	0.768	0.643	0.786	0.432	*0.875*	0.625	0.554	*0.821*	0.696	0.741	0.643	0.696	0.702
S+P50	**1.000**	0.741	0.786	0.696	0.607	*0.864*	0.679	0.625	0.589	0.732	0.786	**0.907**	0.679	0.446	0.724
Averages	*0.882*	*0.816*	*0.899*	*0.886*	*0.833*	0.661	*0.832*	*0.819*	0.569	*0.855*	0.694	*0.849*	0.676	0.754	0.787

Table B.17: AU=27

Training Set	Test Sets													
	Clear	Gaussian	Poisson	Speckle	S+P05	S+P10	S+P15	S+P20	S+P25	S+P30	S+P35	S+P40	S+P45	S+P50	Averages
Clear	**0.996**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.968**	**0.995**	**0.991**	**0.977**	**0.954**	**0.982**	**0.955**	**0.987**
Gaussian	**0.996**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.963**	**0.990**	**0.977**	**0.977**	**0.963**	**0.996**	**0.960**	**0.987**
Poisson	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.972**	**0.990**	**0.991**	**0.972**	**0.958**	**0.978**	**0.955**	**0.987**
Speckle	**0.996**	**1.000**	**0.996**	**0.991**	**1.000**	**1.000**	**0.996**	**0.968**	**0.986**	**0.982**	**0.972**	**0.926**	**0.978**	**0.955**	**0.982**
S+P05	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.986**	**0.986**	**0.986**	**0.986**	**0.977**	**0.996**	**0.951**	**0.991**
S+P10	**0.996**	**1.000**	**0.996**	**0.991**	**1.000**	**1.000**	**0.996**	**0.968**	**0.981**	**0.977**	**0.958**	**0.940**	**0.982**	**0.951**	**0.981**
S+P15	**0.996**	**1.000**	**0.996**	**0.995**	**1.000**	**1.000**	**0.996**	**0.977**	**0.981**	**0.973**	**0.972**	**0.931**	**0.991**	**0.933**	**0.981**
S+P20	**0.986**	**1.000**	**0.982**	**0.986**	**0.996**	**1.000**	**1.000**	**0.963**	**0.976**	**0.977**	**0.982**	**0.921**	**0.991**	**0.946**	**0.979**
S+P25	**0.991**	**1.000**	**0.996**	**0.991**	**1.000**	**1.000**	**1.000**	**0.968**	**0.971**	**0.968**	**0.972**	**0.968**	**0.991**	**0.951**	**0.983**
S+P30	**0.982**	**1.000**	**0.987**	**0.991**	**0.982**	**1.000**	**1.000**	**0.968**	**0.971**	**0.964**	**0.986**	**0.931**	**0.987**	**0.942**	**0.978**
S+P35	**0.991**	**1.000**	**0.982**	**0.991**	**0.996**	**1.000**	**1.000**	**0.968**	**0.986**	**0.973**	**0.963**	**0.940**	**0.991**	**0.942**	**0.980**
S+P40	**0.977**	**1.000**	**0.991**	**0.991**	**0.991**	**1.000**	**0.982**	**0.949**	**0.976**	**0.964**	**0.972**	**0.921**	**1.000**	**0.938**	**0.975**
S+P45	**0.982**	**1.000**	**0.991**	**0.991**	**0.996**	**1.000**	**0.991**	**0.972**	**0.976**	**0.977**	**0.968**	**0.958**	**0.946**	**0.946**	**0.978**
S+P50	**0.986**	**1.000**	**0.960**	**0.991**	**0.978**	**0.995**	**1.000**	**0.944**	**0.986**	**0.964**	**0.977**	**0.963**	**0.973**	**0.915**	**0.974**
Averages	**0.991**	**1.000**	**0.991**	**0.994**	**0.996**	**1.000**	**0.997**	**0.967**	**0.982**	**0.976**	**0.974**	**0.947**	**0.984**	**0.946**	**0.982**

Bibliography

[1] S. Agarwal, A. Awan, and D. Roth, “Learning to detect objects in images via a sparse, part-based representation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 1475–1490, 2004.

[2] B. H. B. Weyrauch, J. Huang and V. Blanz, “Component-based face recognition with 3d morphable models,” in *First IEEE Workshop on Face Processing in Video, Washington, D.C.*, 2004.

[3] M. S. Bartlett, J. C. Hager, P. Ekman, and T. J. Sejnowski, “Measuring facial expressions by computer image analysis,” *Psychophysiology*, vol. 36, no. 2, pp. 253–263, 1999.

[4] M. Bartlett, G. Littlewort, B. Braathen, T. Sejnowski, and J. Movellan, “A prototype for automatic recognition of spontaneous facial actions,” in *Advances in Neural Information Processing Systems, Vol 15. MIT Press - S. Becker and K. Obermayer (eds.)*, 2003.

[5] L. B. Batista and H. M. Gomes, “Photogenic expression recognition using Gabor filters and support vector machines*,” 2005.

[6] A.-A. Bhuiyan and C. H. Liu, “On face recognition using Gabor filters,” in *World Academy of Science, Engineering and Technology 28*, 2007.

[7] W. R. Boukabou, L. Ghouti, and A. Bouridane, “Face recognition using a Gabor filter bank approach,” *Adaptive Hardware and Systems, NASA/ESA Conference on*, vol. 0, pp. 465–468, 2006.

[8] J. M. Buenaposada, E. Mu noz, and L. Baumela, “Efficient-illumination independent appearance-based face tracking,” *Image Vision Comput.*, vol. 27, no. 5, pp. 560–578, 2009.

[9] C. J. C. Burges, “A tutorial on support vector machines for pattern recognition,” *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998. [Online]. Available: citeseer.ist.psu.edu/article/burges98tutorial.html

[10] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[11] G.-C. Chao, S.-S. Lee, H.-C. Lai, and S.-J. Horng, “Embedded fingerprint verification system,” *International Conference on Parallel and Distributed Systems*, vol. 2, pp. 52–57, 2005.

[12] C. I. Chow, S. Member, and C. N. Liu, “Approximating discrete probability distributions with dependence trees,” *IEEE Transactions on Information Theory*, vol. 14, pp. 462–467, 1968.

[13] I. Cohen, N. Sebe, L. Chen, A. Garg, and T. S. Huang, “Facial expression recognition from video sequences: Temporal and static modelling,” in *Computer Vision and Image Understanding*, 2003, pp. 160–187.

[14] J. Cohn, A. Zlochower, J.-J. J. Lien, and T. Kanade, “Automated face analysis by feature point tracking has high concurrent validity with manual FACS coding,” *Psychophysiology*, vol. 36, pp. 35 – 43, 1999.

[15] 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. [Online]. Available: http://www.sciencedirect.com/science/article/B6WCX-45NJT56-1K/2/1933a32b2a89e4c3f032ba0718f4ef03

[16] C. Cortes and M. Mohri, “AUC optimization vs. error rate minimization,” in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. Saul, and B. Scholkopf, Eds. Cambridge, MA: MIT Press, 2004. [Online]. Available: http://citeseer.ist.psu.edu/cortes03auc.html

[17] C. Cortes and V. Vapnik, “Support vector networks,” *Machine Learning*, vol. 20, pp. 273–297, 1995.

[18] M. N. Dailey and G. W. Cottrell, “PCA = Gabor for expression recognition,” in *UCSD CSE TR CS-629*, 1999.

[19] H.-B. Deng, L.-W. Jin, L.-X. Zhen, and J.-C. Huang, “A new facial expression recognition method based on local Gabor filter bank and PCA plus LDA,” 2005.

[20] G. Donato, M. S. Bartlett, J. C. Hager, P. Ekman, and T. J. Sejnowski, “Classifying facial actions,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 10, pp. 974–989, 1999.

[21] B. Druchen, “Deaf Federation of South Africa,” 2009. [Online]. Available: http://www.deafsa.co.za

[22] P. Ekman, “Methods for measuring facial action,” in *Handbook of methods in nonverbal behavior research*, K. Scherer and P. Ekman, Eds. Cambridge: Cambridge University Press, 1982, pp. 45–90.

[23] P. Ekman and W. Friesen, *The Facial Action Coding System: A Technique For The Measurement of Facial Movement*. San Francisco, CA: Consulting Psychologists Press, Inc., 1978.

[24] P. Ekman and E. Rosenberg, “What the face reveals: Basic and applied studies of spontaneous expression using the facial action coding system (FACS),” in *Oxford University Press*, 2004.

[25] P. Ekman, T. Huang, T. Sejnowski, and J. Hager, “Final report to NSF of the planning workshop on facial expression understanding,” National Science Foundation, Human Interaction Lab., UCSF, CA 94143, Tech. Rep., 1993.

[26] P. Ekman, *Telling Lies: Clues to Deceit in the Marketplace, Politics, and Marriage (Revised and Updated Edition)*, 2nd ed. W. W. Norton & Company, 2001. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0393321886

[27] P. Ekman, W. V. Friesen, and J. C. Hager, *Facial Action Coding System (FACS)*. Salt Lake City: A Human Face, 2002.

[28] I. A. Essa and A. P. Pentland, “Coding, analysis, interpretation, and recognition of facial expressions,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 757–763, 1997.

[29] I. Fasel and J. R. Movellan, “Comparison of neurally inspired face detection algorithms, UAM, 2002,” in *Proc. of International Conference on Artificial Neural Networks (ICANN 2002)*, 2002, pp. 1395–1401.

[30] I. R. Fasel and M. S. Bartlett, “A comparison of Gabor filter methods for automatic detection of facial landmarks,” in *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, 2002.

[31] T. Fawcett, “ROC graphs: Notes and practical considerations for researchers,” *Machine Learning*, vol. 31, no. HPL-2003-4, pp. 1–38, 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.10.9777&rep=rep1&type=pdf

[32] R. Fletcher, *Practical methods of optimization*. Chichester: John Wiley & Sons, Inc., 1980, vol. 2.

[33] N. Friedman, D. Geiger, M. Goldszmidt, G. Provan, P. Langley, and P. Smyth, “Bayesian network classifiers,” in *Machine Learning*, 1997, pp. 131–163.

[34] W. Friesen and P. Ekman, “EMFACS-7: Emotional facial action coding system, version 7,” 1984.

[35] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Prentice Hall, 2002.

[36] M. Heller and V. Haynal, “The faces of suicidal depression,” pp. 107–117, 1994.

[37] M. Hsuan Yang, D. Roth, and N. Ahuja, “A SNoW-based face detector,” in *Advances in Neural Information Processing Systems 12*. MIT Press, 2000, pp. 855–861.

[38] C. Izard, L. Dougherty, and E. Hembree, “A system for identifying emotion by holistic judgments (AFFEX),” University of Delaware, Instructional Resource Center, 1983.

[39] C. Izard, *The Maximally Discriminative Facial Movement Coding System (MAX)*, University of Delaware, Instructional Resource Center, Newark, 1979.

[40] 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, 2000, pp. 46–53.

[41] S. Kanfer, *Serious business: the art and commerce of animation in America from Betty Boop to Toy story*. New York: Scribner, 1997, includes bibliographical references and index.

[42] A. Kapoor and R. W. Picard, "Automatic facial action analysis," 2002.

[43] A. C. Koutlas and D. I. Fotiadis, "An automatic region based methodology for facial expression recognition," in *BIOSIGNALS (2)*, 2008, pp. 218–223.

[44] A. Krishan, "Evaluation of Gabor filter parameters for image enhancement and segmentation," Master's thesis, Department of Electrical and Instrumentation Engineering , Thapar University, Jul. 2009.

[45] R. S. Kroon, "Support vector machines, generalization bounds and transduction," Master's thesis, Department of Computer Science, University of Stellenbosch, Dec. 2003.

[46] F. B. Kybernetik, A. Smola, B. Schlkopf, E. Smola, K.-R. Mller, L. Bottou, C. Burges, H. Bulthoff, K. Gegenfurtner, and P. Haffner, "Nonlinear component analysis as a kernel eigenvalue problem," 1998.

[47] A. Lake, "Unicef," 2009. [Online]. Available: http://www.unicef.org

[48] C.-C. Lee, S.-H. Chen, H.-M. Tsai, P.-C. Chung, and Y.-C. Chiang, "Discrimination of liver diseases from CT images based on Gabor filters," *IEEE Symposium on Computer-Based Medical Systems*, vol. 0, pp. 203–206, 2006.

[49] H. Li, J. Buenaposada, and L. Baumela, "Real-time facial expression recognition with illumination-corrected image sequences," 2008, pp. 1–6.

[50] Y. li Tian, T. Kanade, and J. F. Cohn, "Recognizing action units for facial expression analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 97–115, 2001. [Online]. Available: citeseer.ist.psu.edu/article/tian01recognizing.html

[51] J. Lien, J. F. Cohn, T. Kanade, and C. chung Li, "Automated facial expression recognition based on FACS action units," 1998.

[52] G. Littlewort, M. S. Bartlett, and J. R. Movellan, “Are your eyes smiling? Detecting genuine smiles with support vector machines and Gabor wavelets,” in *In Proceedings of the 8th Joint Symposium on Neural Computation*, 2001.

[53] G. Littlewort, I. Fasel, M. S. Bartlett, and J. R. Movellan, “Fully automatic coding of basic expressions from video,” University of California, San Diego, San Diego, CA 92093, INC MPLab Tech Report 3, 2002.

[54] G. Littlewort, M. S. Bartlett, I. R. Fasel, J. Chenu, T. Kanda, H. Ishiguro, and J. R. Movellan, “Towards social robots: Automatic evaluation of human-robot interaction by face detection and expression classification.” pp. –1–1, 2003.

[55] G. Littlewort, I. Fasel, M. S. Bartlett, and J. R. Movellan, “Machine Perceptron Laboratory MPLAB,” 2009. [Online]. Available: http://Markov.ucsd.edu/movellan/mplab

[56] G. C. Littlewort, M. S. Bartlett, and K. Lee, “Automatic coding of facial expressions displayed during posed and genuine pain,” *Image Vision Comput.*, vol. 27, no. 12, pp. 1797–1803, 2009.

[57] W. Liu and Z. Wang, “Facial expression recognition based on fusion of multiple Gabor features,” *International Conference on Pattern Recognition*, vol. 3, pp. 536–539, 2006.

[58] M. Lyons and S. Akamatsu, “Coding facial expressions with Gabor wavelets,” in *Proceedings of the Third International Conference on Face & Gesture Recognition*, Nara, Japan, 1998, pp. 200–205.

[59] S. Marcelja, “Mathematical description of the responses of simple cortical cells,” *J. Opt. Soc. Am.*, vol. 70, no. 11, p. 1297, 1980. [Online]. Available: http://www.opticsinfobase.org/abstract.cfm?URI=josa-70-11-1297

[60] MATLAB, *version 7.10.0 (R2010a)*. Natick, Massachusetts: The MathWorks Inc., 2010.

[61] C. W. Omlin, M. M. Glaser, and J. Connan, “Integration of verbal and signed communication: Real-time sign language translation,” October 2009, `http://cs.uwc.ac.za`.

[62] M. Pantic and L. J. Rothkrantz, “Automatic analysis of facial expressions: The state of the art,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1424–1445, 2000.

[63] P. J. Phillips, H. Wechsler, J. Huang, and P. J. Rauss, “The FERET database and evaluation procedure for face-recognition algorithms,” pp. 295–306, 1998.

[64] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. H. Salesin, “Synthesizing realistic facial expressions from photographs,” in *ACM SIGGRAPH 2006 Courses*, ser. SIGGRAPH ’06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1185657.1185859

[65] H. T. Rosshidi and A. R. Hadi, “Reconfigurable Gabor filter for fingerprint recognition using FPGA verilog,” *AIP Conference Proceedings*, vol. 1136, no. 1, pp. 796–800, 2009. [Online]. Available: http://link.aip.org/link/?APC/1136/796/1

[66] D. Roth, “Learning to resolve natural language ambiguities: a unified approach,” in *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, ser. AAAI ’98/IAAI ’98. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1998, pp. 806–813. [Online]. Available: http://portal.acm.org/citation.cfm?id=295240.295894

[67] P. RothKrantz, “Automatic Analysis of Facial Expressions: the state of the art,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1424–1445, 2000.

[68] R. Sandler and M. Lindenbaum, “Gabor filter analysis for texture segmentation,” in *Conference on Computer Vision and Pattern Recognition Workshop (CVPRW06)*, 2006.

[69] T. Sim, S. Baker, and M. Bsat, “The CMU pose, illumination, and expression (PIE) database,” in *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, May 2002.

[70] H. Tao and T. S. Huang, “Connected vibrations: A modal analysis approach for non-rigid motion tracking,” in *Computer Vision and Pattern Recognition*, 1998, pp. 735–740.

[71] M. Ursino, G.-E. La Cara, and M. Ritrovato, “Direction selectivity of simple cells in the primary visual cortex: Comparison of two alternative mathematical models. i: Response to drifting gratings,” *Comput. Biol. Med.*, vol. 37, pp. 398–414, March 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1223925.1224148

[72] V. N. Vapnik, *Statistical Learning Theory (Adaptive and Learning Systems for Signal Processing, Communications, and Control)*, Jun. 1998.

[73] P. Viola and M. Jones, “Robust real-time object detection,” in *International Journal of Computer Vision*, 2001.

[74] T. D. B. W. M. Pan, C. Y. Suen, “Script identification using steerable Gabor filters,” in *First NASA/ESA Conference on Adaptive Hardware and Systems*, 2006.

[75] W. Wang, J. Li, F. Huang, and H. Feng, “Design and implementation of log-Gabor filter in fingerprint image enhancement,” *Pattern Recogn. Lett.*, vol. 29, pp. 301–308, February 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1326360.1326443

[76] J. Weston and C. Watkins, *Support vector machines for multi-class pattern recognition*. In Proceedings of the Seventh European Symposium On Artificial Neural Networks, 1999, vol. 4, no. 6, pp. 219–224. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.144.1692

[77] J. Whitehill and C. Omlin, "Local versus global segmentation for facial expression recognition," in *FGR 2006: Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition*, Washington, DC, USA, 2006, pp. 357–362.

[78] J. Whitehill and C. W. Omlin, "Haar features for FACS AU recognition," in *FGR '06: Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 97–101.

[79] J. R. Whitehill, "Automatic real-time facial expression recognition for signed language translation," Master's thesis, Department of Computer Science, University of the Western Cape, May 2006.

[80] Y.-T. Wu, T. Kanade, J. Cohn, and C.-C. Li, "Optical flow estimation using wavelet motion model," in *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*. Washington, DC, USA: IEEE Computer Society, 1998, p. 992.

[81] D.-D. Yang, L.-W. Jin, J.-X. Yin, L.-X. Zhen, and J.-C. Huang, "Facial expression recognition with pyramid Gabor features and complete kernel Fisher linear discriminant analysis."

[82] P. Yao, J. Li, X. Ye, Z. Zhuang, and B. Li, "Iris recognition algorithm using modified log-Gabor filters," in *Proceedings of the 18th International Conference on Pattern Recognition - Volume 04*, ser. ICPR '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 461–464. [Online]. Available: http://dx.doi.org/10.1109/ICPR.2006.726

[83] Z. Zhang and Z. Zhang, "Feature-based facial expression recognition: Sensitivity analysis and experiments with a multilayer perceptron," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 13, pp. 893–911, 1999.