The Application of Evolutionary Algorithms to the Classification of Emotion from Facial Expressions

Ce Shi

Doctor of Philosophy

Electronics

University of York

October 2014

Abstract

Emotions are an integral part of human daily life as they can influence behaviour. A reliable emotion detection system may help people in varied things, such as social contact, health care and gaming experience. Emotions can often be identified by facial expressions, but this can be difficult to achieve reliably as people are different and a person can mask or supress an expression. Instead of analysis on static image, the computing of the motion of an expression's occurrence plays more important role for these reasons. The work described in this thesis considers an automated and objective approach to recognition of facial expressions using extracted optical flow, which may be a reliable alternative to human interpretation. The Farneback's fast estimation has been used for the dense optical flow extraction. Evolutionary algorithms, inspired by Darwinian evolution, have been shown to perform well on complex, nonlinear datasets and are considered for the basis of this automated approach. Specifically, Cartesian Genetic Programming (CGP) is implemented, which can find computer programme that approaches user-defined tasks by the evolution of solutions, and modified to work as a classifier for the analysis of extracted flow data. Its performance compared with Support Vector Machine (SVM), which has been widely used in expression recognition problem, on a range of pre-recorded facial expressions obtained from two separate databases (MMI and FG-NET). CGP was shown flexible to optimise in the experiments: the imbalanced data classification problem is sharply reduced by applying an Area under Curve (AUC) based fitness function. Results presented suggest that CGP is capable to achieve better performance than SVM. An automatic expression recognition system has also been implemented based on the method described in the thesis. The future work is to propose investigation of an ensemble classifier implementing both CGP and SVM.

Contents

List of Figures

List of Tables

Acknowledgments

Firstly I would like to thank to Dr. Stephen Smith, my PhD supervisor. Without his help and guidance I could not go this far.

I would also thank to my wife and my parents. Thank you for the support and understanding.

Special thanks to the candidates and friends who help me to collect expression videos.

Finally I would thank to all my friends, thank you for the help and encouragement.

Declaration

I declare I am the sole author of the thesis titled ‘The Application of Evolutionary Algorithms to the Classification of Emotion from Facial Expressions’. The thesis does not contain materials that have been presented before in published papers or papers under review. This work has not previously been presented for an award at this, or any other, University.

1 Introduction

1.1 Emotions and Expressions

Emotions are very important for humans in their daily lives. People's behaviour can be influenced by emotions. With an angry emotion, we may do something that we would not normally do. When we feel sad, we want somebody to talk to. The decisions we make are greatly affected by our current emotions: they will be different when we feel happy, sad, angry or fear. It is obviously important for others as well as the individual to recognise these emotions, if the consequences may be far reaching.

Affective Computing[1] is the concept that the computer has the intelligent to understand, recognise and interpret human emotions. The applications can be useful in (many) areas and possibly changing (the industry). An application in game industry will not only wait the intentional facial expression as an input method or a desired appearance to perform, but also detect the real emotion of the player during the game and the game process or ending may change based on it in order to bring the player special experience. In daily life, a wearable device that can read emotion during communication will help people has problem to sense other's affects or even give expert's advice to less experience people, for example, the notice of unnatural or suspect expressions. Such device can be also used for monitoring the user's emotion. It can aware the user when he/she is in an extreme emotion, in case he/she would do regretful things. The everyday emotion data may be also saved together with other data, like heart rate and blood pressure, for healthcare.

In order to begin to apply technology to this problem, we first need to understand some of the underlying theory of emotion in more fully. In Scherer's Component Process Model of emotion[2], five crucial elements of emotion are identified: (i) an evaluation of events and objects – the cognitive component of appraisal; (ii) system regulation – the neuropsychological component of bodily symptoms of emotional experience; (iii) preparation and direction – which is a motivational component for the preparation and direction of motor responses; (iv) communication of reaction and behavioural intention expression – in which facial and vocal expressions almost always accompanies an emotional state; and (v) monitoring of internal state and organism–environment interaction - feelings or the subjective experience of emotional state once it has occurred. For many of us facial expression is one of the most recognisable elements of emotion. We read facial expression to know one's emotion and predict the behaviour under that emotion. This is actually the purpose of emotion as Charles Darwin explained it in his book "The Expression of the

Emotions in Man and Animals" [3]. Darwin argued that emotions served a purpose for humans and other mammals, in communication and also in aiding their survival.

How do facial expressions happen? There are two brain pathways associated with facial expression; the first is voluntary expression, which is made consciously. The second type of expression is emotional, which is often made unconsciously[4]. Either path controls the muscles on face to contract or expand. The movement of the muscles also moves the skin that connects to them. This causes the changes of appearances of one's face. People read expressions from the appearance or the movement of the face to know one's emotion[5].

Are expressions universal? People have many expressions and emotions but do different people have same type of expression for one emotion? The universality of expressions was first suggested by Darwin in 1872[3], under his theory of evolution. After much subsequent research it has been suggested that several expressions are universal among people, even when living in different cultures and environments[6]. This universality of expressions provides us with a basis for expression recognition.

1.1.1 Differences in Individuals' Expressions

Although many accept expressions are universal, there are still some differences between people and in specific cases. People have different appearances, and even though they use the same muscles for forming an expression, it may look different. Equally, for just one individual, it is unlikely they will have exactly the same expression for one emotion all the time. First of all, the strength of expression may be different for same emotion. Secondly, in different environments, situations and conditions, one may act differently. For example, our expression when talk with someone familiar may be different from when we talk with a stranger, for the same emotion. Furthermore, there are more complicated cases that make the expressions vary. People may intentionally fake expressions for the purpose of hiding real emotion. For example, we have all experienced someone with a fake smile to prevent others worrying, when there is really sadness and reluctance to share the reason. One may pretend to be sad when listening to another's misfortune, but actually this emotion may not necessarily be shared. As mentioned above, the expressions controlled by the second brain pathway will occur unconsciously, so making another, different expression to hide the real emotion will cause a subtle change on face appearance. These are called micro expressions[7], and can occur in a very short time, and so are easy to miss. However, they provide a very useful ability to differentiate expressions and underlying emotions.

1.2 Automatic Recognition of Human Expression

Automatic expression recognition has always been a popular problem in image processing and has attracted a great amount of research. There are many potential benefits of such systems as automated recognition of human facial expressions by computer can reduce the workload of humans. Machines can work continuously without rest and variability, unlike their human counterparts. Not only do automated systems offer objectivity, reliability and reproducibility of results, but can also discover information that may be difficult or impossible to observe with the human eyes. With the rapidly developing technology, high-resolution cameras are inexpensive and commonly available, as is the supporting software and related techniques for processing of images and video. This makes such complex tasks such as recognising human expression more practical and realisable.

There is currently a number of approaches that are used to recognise facial expressions automatically, including the analysis of facial appearance[8], and location and tracking of landmarks[9] and flows[10] of the face for extracting facial information. The information gained from these techniques can then be used to train a decision making machine to identify or *classify* the expression detected. First of all, a predetermined set of data is entered into the machine together with appropriate class labels describing the expressions exhibited (such as fear or happiness). The machine will analyse the input data and labels and is trained develop the knowledge of finding relations between data and different labels. After this has been accomplished, the machine will give its response based on the pre-trained experience to new data input. This process is called machine learning.

Many types of classifiers have been applied to expression recognition with some success, such as Support Vector Machine[11-15], Bayesian Networks[16-18] and Neural Networks[19-22]. But Evolutionary Algorithms[74-92], which have been shown to be effective for other classification applications, have yet to be investigated for expression recognition.

1.3 Evolutionary Algorithms

Evolutionary algorithms (EAs) are stochastic search techniques inspired by Darwin's theory of evolution[3]. Throughout history, individuals who are best suited to the environment are most likely to survive and produce the next generation. Their offspring also inherits traits that will propagate this fitness; termed *natural selection*, it will continue to influence future generations.

This repeated process leads to increased fitness, permitting the population to become stronger and survive in the environment. Evolutionary algorithms simulate this process with the purpose of finding the best individual of solutions for a particular problem, in this case a classifier for recognising expressions.

1.3.1 Advantages of Evolutionary Algorithms

Due to their specific characteristics, EAs can be more effective than conventional, statistically based approaches for solving highly complex, non-linear and discontinuous search problems[23]. Consequently, they are gaining considerable popularity, not least of all, because they can solve problems requiring multiple solutions and are well suited to parallel processing[24].

1.4 Hypothesis

The potential benefits of automating the recognition and classification of facial expressions and the implementation of an evolutionary algorithm to achieve this forms the basis of this thesis and has been summarised in the following hypothesis:

“Evolutionary Algorithms are an effective means of recognising and classifying human facial expression.”

EAs have been chosen as they have been shown to have several distinct advantages over other learning algorithms. Their application to the classification of facial expressions will be compared with SVM - a popular, competitive classifier. Sample expressions will be taken from several video databases.

1.5 Thesis Structure

The structure of this thesis is based on the routine of the research on expression recognition and evolutionary algorithms. It starts at the literature reviews on the background of image processing on expression recognition and Evolutionary Algorithms. Suitable image processing techniques for the facial expression recognition problem are studied. Then the method used for extracting features is applied on both conventional classifier and Evolutionary Algorithm based classifier. In the end, the comparison of the two classifiers (SVM and CGP) is made.

Chapter 2 introduces the background of facial expression recognition and evolutionary algorithms. In 2.1, literatures of past works on this problem are reviewed and important methods and techniques are introduced. In 2.2 Evolutionary Algorithms are introduced including the concept, structure and algorithms. Then Genetic Programming is on focus, the algorithms are discussed and related literatures are reviewed.

Chapter 3 is mainly discussing how the feature is extracted for the expression recognition. Firstly the database is introduced in 3.2. In 3.3 the details of Optical Flow algorithm are studied. The video data will be pre-processed and the method is in 3.4. Then three ways of reduction on extract Optical Flow feature are discussed. The comparison is made by the classification results with SVM classifier.

Chapter 4 introduces the CGP and experiments on recognition by using the features from chapter 3 for a comparison to SVM. The CGP structure and evolutionary strategy are described in 4.2. Then the classifier based on CGP is introduced in 4.3 together with a series of experiments and all the experimental results of CGP classifier are compared with those of SVM. This CGP classifier is applied to the expression flow data from chapter 3 in 4.4. In 4.5 the fitness function is discussed and the AUC based fitness function is applied to improve the recognition rates. In 4.6 the overfitting of the classifier training is discussed and tried to avoid by adjusting the CGP classifier. Then this optimised CGP classifier is compared with the SVM classifier by the classification results on the facial expression recognition. In 4.7 an introduction of the implementation of the automatic recognition system is made.

Chapter 5 is the conclusion. It summarises the works have been done and discoveries from all the experiments in this research. The conclusions and recommendations for future work are given in this chapter.

2 Background

2.1 Previous Work on Expression Recognition

Facial expression recognition is an important topic in image processing, and has attracted considerable attention from the research community. To evaluate an expression recognition method, the speed and accuracy should be considered, but in the literature, many different databases have been used and so experimental results cannot be easily compared using different technologies. Some methods also require manual land marking of feature points, making a direct comparison unfair. Additionally, the difference between using static image data and dynamic image data is also a factor that makes comparison unrealistic. Finally, only a few papers mention the processing speed of their systems when reporting system performance.

A typical expression recognition system consists of three procedures: (i) face detection, (ii) feature extraction and (iii) classification. Face detection is essential to locate the appropriate area of the image on which pre-processing and feature extraction can be undertaken ahead of the recognition phase. The data representing a face image is usually too large for further immediate processing and feature extraction is used to reduce the dimensionality of data to be processed. Classification is used to identify which category the input features (or expression in this case) belong to. If the features are too many, the learning process of the classifier becomes computationally and analytically demanding. This often results in a rise of classification error, because of the interference of less relative data or even noise. Hence, feature extraction and selection are very important to reduce the dimensionality of the feature space before classification[25].

2.1.1 Face Detection

Face detection is the very first step of processing and used to acquire the location of the face in the image or video frame being analysed and, as such, is an essential part of expression recognition. Many face detection techniques have been proposed which are often integrated with feature extraction in order to save computation time. Since Viola and Jones proposed their robust and fast face detection method in 2002, it has become possible to build real-time facial expression recognition systems in practical, typically achieving a speed of 25 Frames per Second (FPS) on a Pentium III computer[26]. Consequently, the Viola and Jones' algorithm has become one of the most widely used face detection techniques and even today attracts much attention from researchers. T. Ephraim et al. optimised the Viola and Jones' algorithm to make it suitable

for working from within a web browser[27]. L. Lang and W. Gu improved the original algorithm that avoids the over training phenomenon occurred in original algorithm[28]. R. Liu et al. applied a continuously adaptive mean shift algorithm to track the face after detection by the Viola and Jones' algorithm[29]. O. Bilaniuk et al. applied a fast face detection method on Single Instruction Multiple Data architectures replacing the Haar feature by Local Binary Patterns feature[30].

2.1.1.1 The Viola and Jones' Algorithm

The Viola and Jones algorithm applies a cascade AdaBoost learning algorithm to select Haar-like features in the detection of faces.

Haar features consider the difference between two or more adjacent rectangles within an image. The rectangles can be of any size and any position within the searching window, but a Haar feature only considers the adjacent rectangles which are of the same size, as shown in figure 2.1. The rectangles can comprise of two to four regions and they can be adjacent horizontally or vertically with the same size and shape. The Haar feature computes the sum of the pixels' grey-scale values in white rectangles subtracted from the sum of those in dark rectangles. The *integral image*[26], which is the summation of the pixels above and to the left of one point in the image, is used for computing the value of all the pixels within a rectangle based on four vertices. For example, in the figure 2.1, the sum of pixels in dark rectangle in B and B' can be calculated from four integral images of points 1, 2, 3 and 4, which is equal to:

integral image 4 - integral image 2 - integral image 3 + integral image 1.

Similarly, in B and B', the sum of pixels in white rectangle is equal to:

integral image 6 - integral image 4 - integral image 5 + integral image 3.

Because each rectangular area in a feature is always adjacent to at least one other rectangle, it follows that any two-rectangle feature can be computed in six array references, any three-rectangle feature in eight, and any four-rectangle feature in just nine. Although the Haar feature is a simple integral and subtraction of arrays, the number of features is too large. A 24 by 24 pixel scan window comprises 576 pixels, but 45,396 possible features and, consequently, the AdaBoost algorithm is used to select useful features in the training phase.

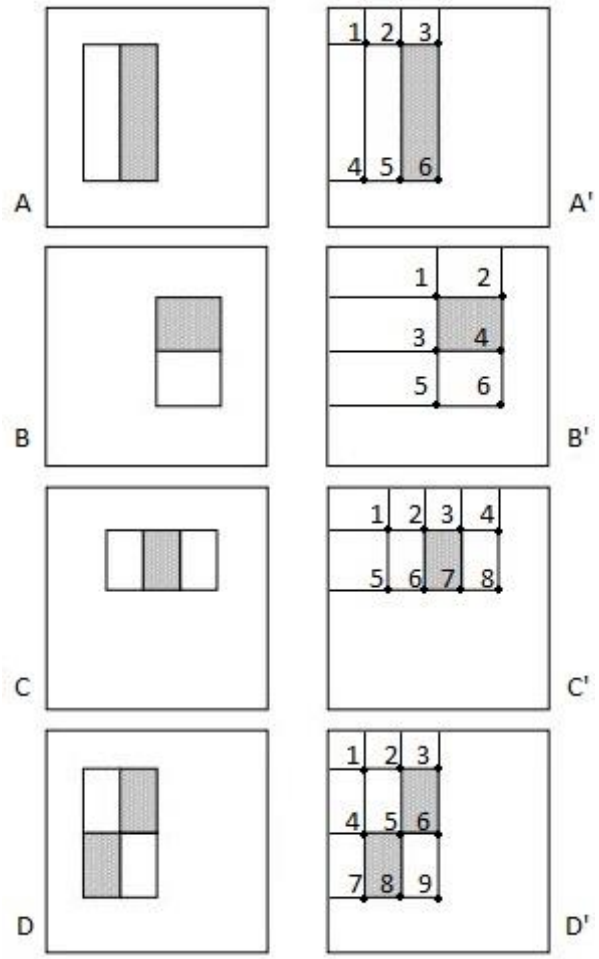


Figure 2.1: Haar features in original Viola and Jones algorithm[26] and the integral images for computing the features

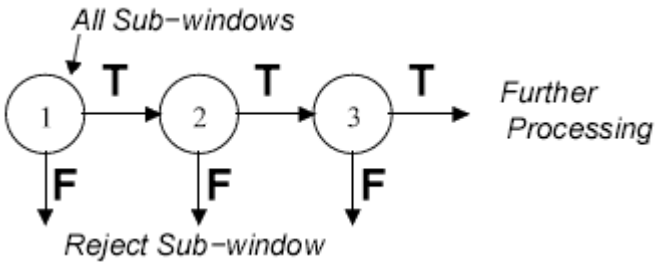


Figure 2.2: Architecture of the cascade classifier[26]

A predetermined number of features is selected and weighted according to the error obtained when each feature is used for detecting an object. Each feature is considered a weak classifier but, in combination, all features form a strong classifier. However, this method is not considered fast enough by the authors and they propose a cascade architecture which increases the classification speed to work in real-time. This architecture cascades several levels of weak classifiers together,

as shown in figure 2.2. If the input passes all the levels, the classifier will give a positive answer; if any weak classifier gives a negative result, the input will not go through to the next level and the classifier will process a new input from the next scan window. Every level should let all the face images pass, but have a high false positive rate as a consequence. The features in next level are always stronger than the previous level and can distinguish more difficult cases which upper levels cannot do. Even though there is a high false positive rate in each weak classifier, a strong classifier can still be achieved at a very low false positive rate, due to the cascade structure.

2.1.2 Feature Extraction

Feature extraction is a very important component of the recognition process and involves extracting the most important information for discriminating different classes. The process reduces the dimensionality of the feature data and can help classifiers to have better performance. There are various types of feature extraction techniques that have been applied for expression recognition, some of which are considered below.

Ekman and Friesen developed the *Facial Action Coding System* (FACS)[31]. This encodes a single muscle or a group of muscles from the face to 46 predefined action units. It is proposed that all facial movements can be described by the combination of these action units, and further, it is suggested that through analysing these action units, psychologists can not only determine a particular facial expression, but also the state of mind or emotion of the subject. Several automated implementations of FACS have been proposed that exploit facial action units to analyse facial expressions[8, 16, 18, 32-34].

John Robinson proposed covariance matrix estimation for face description[35-37]. G. R. S. Murthy and R. S. Jadon introduced an expression recognition method based on *eigenfaces* – the term given to a set of eigenvectors when used for human face recognition[38]. The eigenface technique is still sensitive to shape changes, especially when used for rigid object recognition, but the technique is popular with many researches concerned with analysing human emotion through the use of multiple modalities[39-41]. There are also some approaches for expression recognition using a 3D model[42-46]. I. Cohen et al and A. Azcarate et al proposed two expression recognition systems[44], both of which were based on the Piecewise Bezier Volume tracker[43], increasing the performance of the recognition system by using manifold learning[17, 47, 48].

In addition to these methods, Gabor Filters[49-52], Local Binary Patterns[12-14, 53], Active Appearance Model[19, 32, 54, 55] and Optical Flow[9-11, 56] are amongst the most popular feature extraction techniques employed. The following sections consider these feature extraction algorithms in some detail.

2.1.2.1 Gabor Filters

A Gabor filter[49], named after Dennis Gabor, is a linear filter used for edge detection. Frequency and orientation representations of Gabor filters are similar to those of the human visual system, and they have been found to be particularly appropriate for texture representation and discrimination. In the spatial domain, a 2D Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave. The complex sinusoid is known as the carrier and Gaussian function is known as the *envelope*. The Gabor filters are self-similar: all filters can be generated from one mother wavelet by dilation and rotation.

A 2D Gabor kernel can be mathematically defined as (real part):

$$G(x,y) = \exp(-\frac{x'^2+y'^2}{2\sigma^2})\cos(\frac{2\pi}{\lambda}x') \qquad (2.1)$$

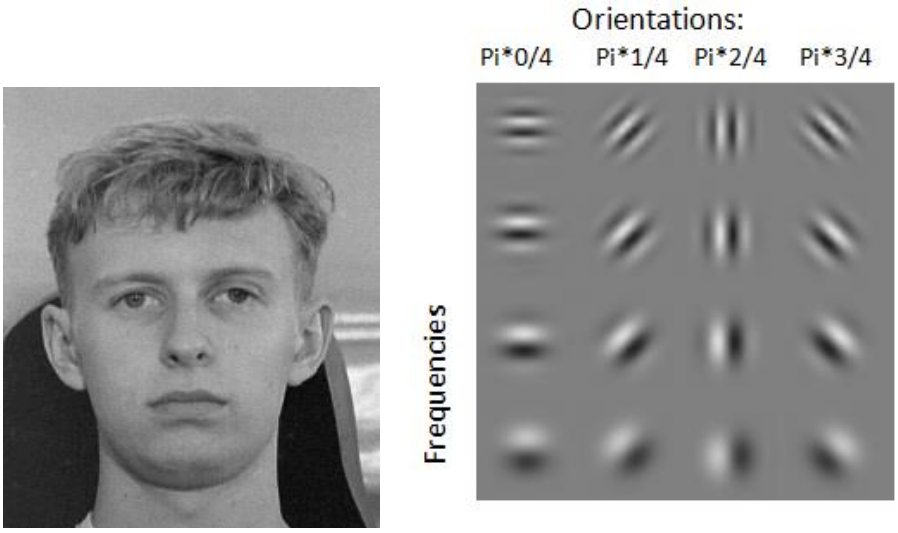
where

$$x' = x\cos\theta + y\sin\theta \qquad (2.2)$$

$$y' = -x\sin\theta + y\cos\theta \qquad (2.3)$$

and λ is the wavelength of the sinusoidal function, θ is the orientation of the Gaussian function, σ is the standard deviation of the Gaussian envelope.

An example transformation from the original image to a Gabor kernel bank is shown in figure 2.3. The result after Gabor filtering is shown in figure 2.4.



(a) (b)

Figure 2.3: (a) Original face image (b) Gabor kernel bank

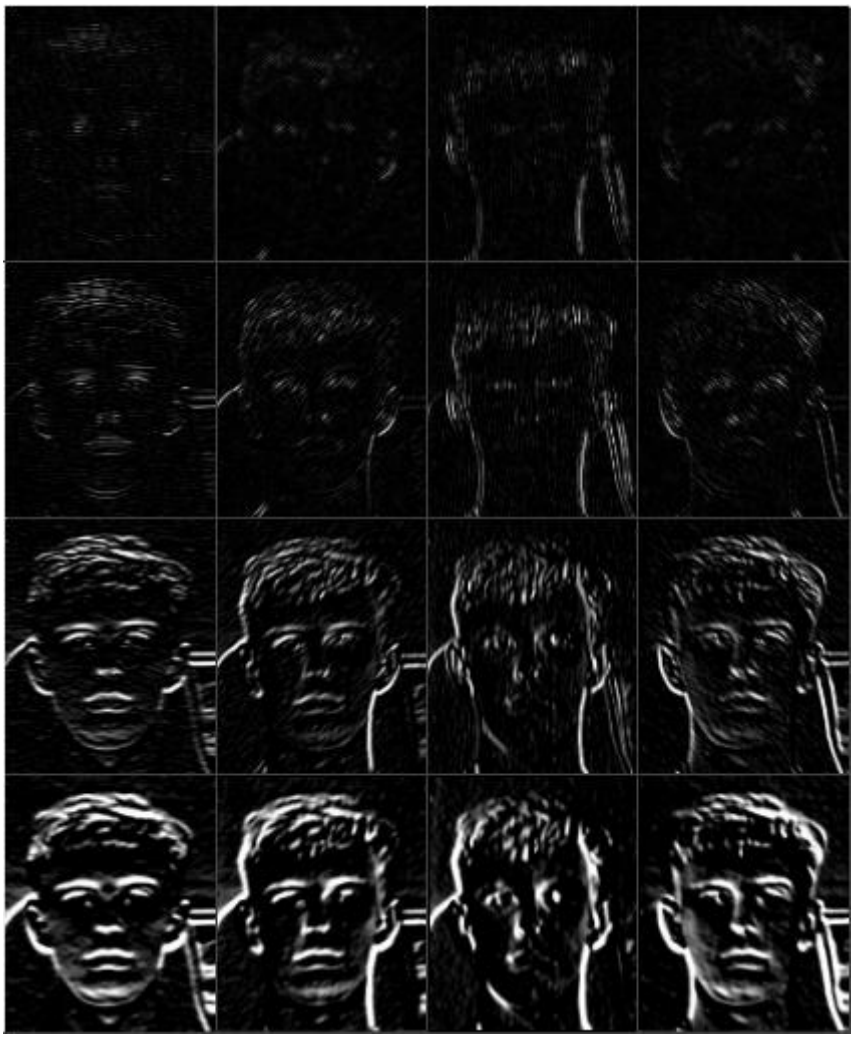


Figure 2.4: Results after applying Garbor filter

The Gabor wavelet is a useful technique in image processing and especially strong for rigid object recognition, but it can also be used for facial expression recognition. I. R. Fasel and M. S. Bartlett applied the Gabor filter for automatically marking pupils and the philtrum (the vertical groove in the middle of the upper lip) in face images[49]. B. Oshidari and B. N. Araabi proposed an adaptive Gabor wavelet for expression recognition[50]. They applied a fuzzy controller in the system to determine the parameters of Gabor wavelet. P. Wu et al introduced an expression recognition system combining the Gabor feature and Active Shape Model (ASM)[51]. Firstly, the ASM was fitted to the face in the image, and then the features of the ASM points were extracted using a Gabor wavelet. The final feature set, a combination of the two (ASM and Gabor feature) was then used for recognition. In a further development, J. Yu and B. Bhanu applied genetic programming to synthesise new features based on Gabor wavelets for expression recognition[52].

2.1.2.2 Local Binary Patterns

The original concept of Local Binary Patterns (LBP) was introduced by Ojala in 1994 for texture analysis [57]. This method compares one pixel in an image with all 8 surrounding pixels. Each surrounding pixel which is smaller than the centre pixel is given a value of "0" at its position, otherwise it is given a value of "1". Following this, an 8-bit binary number is obtained by following a clockwise sequence from the upper-left corner and forms the *LBP code*. The resulting 256-bin histogram using this code describes the texture of the image. The original LBP only used 3 by 3 neighbourhood pixel windows, but later LBP implementations extended this to calculate the pixels of a specified radius. Indeed, the number of pixels and the radius can be set to meet the application requirement. Instead of just describing the texture of the surrounding pixels, this improved method can describe the relationship between more distance pixels within the image, as shown in figure 2.5.

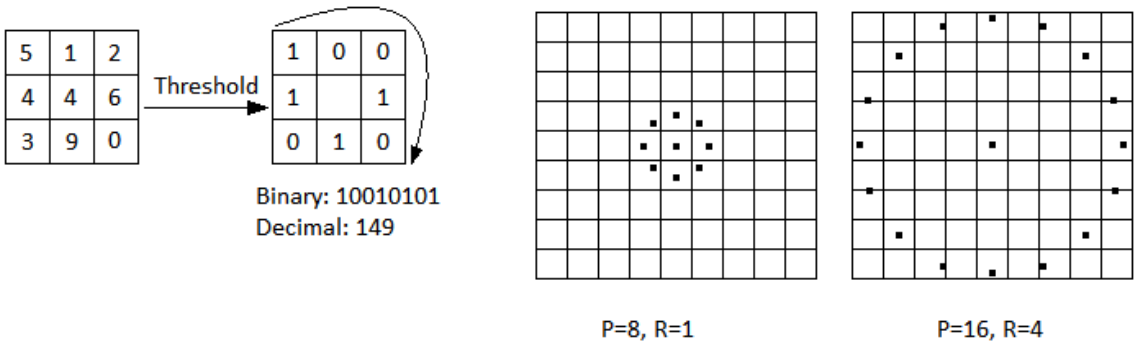


Figure 2.5: Local binary pattern (LPB) algorithm (based on[12])

Where P stands for the number of points, R is the radius of the circle of points.

The number of possible LBP values in an 8 point situation is 2^8 or 256. It has been demonstrated by Ojala et al.[57] that some patterns carry more useful information than others and so LBP only uses these patterns to describe texture and are named *uniform patterns*. Consider the very left and the right bits are connected in the 8 bits string as they are from the circle around the centre pixel. Then if there are two transitions of bit value in the string (value of one bit is not equal to the bit on the right/left), this pattern is called uniform pattern. In the 8 points string, there are 58 uniform patterns of all 256 patterns. Each uniform pattern is in a unique bin and the rest 198 patterns are in one bin, so there are totally 59 bins for the LBP histogram.

This LBP histogram contains information about the distribution of the local micro-patterns, such as edges, spots and flat areas, over the whole image, so can be used to statistically describe image characteristics. But the histogram does not indicate any information about their locations. To overcome this problem, the image is divided into equal regions and histogram of each region is used for texture analysis.

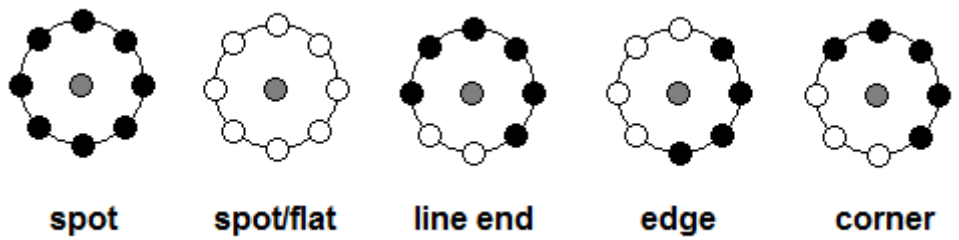


Figure 2.6: Textures describe in LBP[53]

Figure 2.6 shows how LBP describes the textures. The line end, edge and corner textures in the figure are the uniform patterns in LBP.

LPB is a texture description technique. Compared with the Gabor wavelet, it requires less computation time. Caifeng Shan et al applied LBP in their facial expression recognition system[12]. They divided face images, which are normalised by fixing the distance between eyes into 6 by 7 sub-regions and connected LBP histograms of each sub-region together as the feature vector. These sub-regions had their corresponding weights for classification. Subsequently, they introduced a method that selected features by Adaboost[53]. It was different from the method in [12], sub-regions that were extracted by shifting and scaling a sub-window of 16,640 pixels. Adaboost was then used for selecting 50 histograms, which included more discriminating information. G. Zhao and M. Pietikainen proposed two LBP based dynamic texture extraction techniques for expression recognition: *volume local binary patterns* and *local binary patterns* from three orthogonal planes[13]. These two LBP based methods calculated the LBP value not

only in current (single) image but include previous and posterior frames to describe the dynamic texture. Ruo Du et al. proposed Histogram Variance Face (HVF)[14] which is an LBP-based method including temporal information for expression recognition.

2.1.2.3 Active Shape and Active Appearance Models

The *Active Shape Model* (ASM), proposed by T. Coots[58], is used for representing the shape of face by the position of a group of points and the curves they describe.

First, the contours of all the features on the image of a face are marked by hand using a set of landmark points. These labelled face images are used as a training set. In every training image, the numbers of points are same and the position that each point should locate at. A vector ***X*** is formed using the position of all the points whose number and order are confirmed in one image as shown below:

$$X = [x_1, x_2, x_3 \ldots x_n, y_1, y_2, y_3 \ldots y_n] \quad (2.4)$$

Where n is the number of points, x and y are the pixel coordinates. The rotation and scale of the shape should be normalised to minimise the square error of points. Then Principal Component Analysis[17] is applied to all the vectors to describe the majority variations. Only a few eigenvectors remain which correspond to higher eigenvalues. Then a statistical shape model called Point Distribution Model(PDM)[58] is:

$$X = \bar{x} + P_s * b_s \quad (2.5)$$

Where $P_s = (p_1, p_2, p_3 \ldots p_t)$ is the matrix of remaining t eigenvectors, $b_s = (b_1, b_2, b_3 \ldots b_t)$ is the weights. New models can be generated by changing the parameters in b_s. The range of parameters in b_s is learnt from the training set. $\bar{x}$ is the mean shape of the training data.

When a new image is considered, the model is required to fit the new face by changing the shape and pose parameters. The pose parameters include rotation and scale, the shape is changed by adjusting the parameters in b_s. First, a very rough initial parameter is set, then a better parameter is obtained and updated, if found. This is repeated until it converges. There are several methods to achieve this - one is to find edges along the normals through each point and then move the points towards the edges, as this is where the points are supposed to sit.

Texture information is added to the ASM points to form an *Active Appearance Model* (AAM). Proposed by Coots et al.[59], AAM can synthesis new face images, not only the shape points, but also the appearance. The way texture information is added is similar as that used in the PDM.

This technique can match the appearance and shape model to a new image from training data. H. Choi and S. Oh introduced AAM with efficient second order minimisation for face identification and expression recognition[54]. This method avoids a large error in classification, but is computationally intensive. L. Zalewski and S. Gong extended the AAM and proposed mixture of probabilistic and principal component analysis algorithms for expression recognition[55]. This method can generate side views automatically from near frontal images. N. Neggaz et al. proposed a facial expression recognition system based on improved AAM[19]. They applied a differential evolution algorithm to optimise the AAM parameters. A. Ryan et al. gave a definition of constrained local models which included AAM and discussed automated facial expression recognition systems[32]. J. Sung et al. proposed a stereo active appearance model for expression analysis[60]. This method required two cameras in a fixed angle to get the input data. It increased the accuracy and speed of face matching over the original AAM. T. Robin et al. proposed a discrete choice model based method for expression recognition[61]. They applied AAM in their method for feature extraction.

2.1.2.4 Optical Flow

Optical flow is a popular computer vision technique and commonly used for motion detection, object tracking and object segmentation[62]. It describes the movement of an object by the velocities of points on its surfaces and edges, projected on a scene between the observer and the object. In a practical case, it is used for estimating the velocity of a point moving in two consecutive video frames. With these velocities known, the movement of an object, or observer, in image sequences can be calculated. For example, points with similar velocity can be considered as a moving object. i.e. objects can be tracked by calculating the velocities of the points on the object.

Optical flow assumes that the intensity of pixels do not change while the pixel moves through frames, where intensity means the colour or grey level value of a pixel. Therefore, the intensity of a pixel in location (*x, y*) at time t and in a different location (*x+*Δ*x, y+*Δ*y*) after Δ*t* time, can be described as:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \qquad (2.6)$$

Where I(x, y, t) represents the intensity of the pixel in *x,y* at time *t*.

An expansion of this equation can be written in terms of Taylor's series:

$$I(x+\Delta x, y+\Delta y, t+\Delta t) = I(x,y,t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + \ldots \quad (2.7)$$

Assuming the movement is small, so that the second and higher order terms can be ignored, from equations 2.6 and 2.7 we can arrive at the following:

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0. \quad (2.8)$$

Multiply $1/\Delta t$ at both side of the equation produces:

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t}\frac{\Delta t}{\Delta t} = 0 \quad (2.9)$$

Then $\Delta x/\Delta t$ can be presented by V_x means the x component of velocity and $\Delta y/\Delta t$ is shown as V_y:

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0 \quad (2.10)$$

Where $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are the derivatives of the image at (x,y,t) in the corresponding directions. In equation 2.10 there are two unknowns which cannot be solved. As shown in figure 2.7, we cannot tell the displacement is on x, y or both axes from the shift of the line. This problem in an optical flow algorithm context is called the *aperture problem*. An additional set of equations is required for solving this problem which is generated by applying additional constraints. All optical flow algorithms have the possibility of additional conditions which provide constraint equations. The Lucas-Kanade (LK) method[63] and Horn-Shrunk method[62] are used most frequently. The Lucas-Kanade method assumes that the locations of pixels in a small region do not change or that their displacement is small between two adjacent frames, making this a suitable method for object tracking.

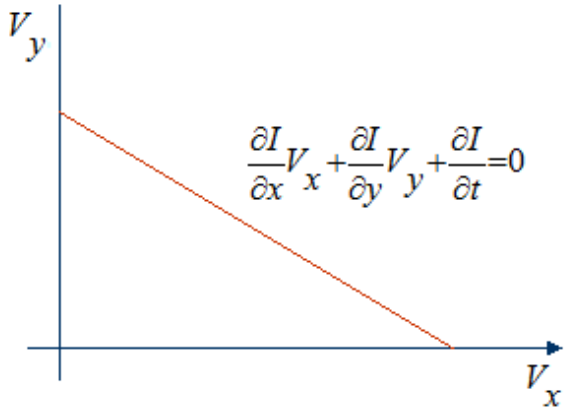


Figure 2.7: The motion constraint line[62]

Optical flow is well known technique for motion estimation and tracking. Kun He et al applied Lucas-Kanade-Tomasi optical flow for the facial feature tracking[56]. They aligned a set of points at carefully selected positions according to the facial action coding system (FACS) on the first frame of the input stream as the features to track. Ren C. Luo et al. proposed a real-time facial feature tracking method based on AAM and optical flow[9] and Byungsung Lee et al. proposed a facial expression classification method[11]. They applied the Lucau-Kanade optical flow in their system to track 18 feature points on the face. Then the traces were used for classifying expressions by support vector machine. X. Peng et al. applied Horn-Schunck optical flow for expression recognition[64]. B. K. Dehkordi and J. Haddadnia proposed a method for facial expression recognition[10]. They characterized the main components on the face by using a Gabor filter on the first frame. Then Iterative LK algorithm was applied and the first 1000 vectors of points in which movements were bigger than others were extracted for classification. Kong Jian et al. introduced a system for expression recognition[65]. They applied two different methods for feature extraction in their method. Volume Local Binary Pattern was used to extract the features of eye region and LK optical flow was used to track the automatically detected feature points of mouth region. G. Wang applied a novel optical flow algorithm with the additional constrains of first-order and second-order div-curl splines for expression recognition[66]. K. Anderson and P. W. McOwan proposed an expression recognition system in which optical flow was used for feature extraction[67]. They integrated a modified ratio template algorithm based face tracker, the multichannel gradient model for optical flow estimation and Support Vector Machines for expression classification. M. Shreve et al. introduced a method for detecting macro- and micro-expression by analysing optical flow based strain patterns[68]. Chao-Kuei Hsieh et al applied optical flow algorithm for cancelling the variations caused by expressions in their face recognition system[69].

2.1.3 Classification

Classification in machine learning is the process that assigns new input data into different categories or classes based on the analyses of the previously provided training data. Usually, instead of the raw data, the features extracted in the previous section are the inputs of the classifier. There are two modes for a classifier: training mode and testing mode. Before a classifier is applied to previously unseen data, it must be trained. The feature vectors combined with expression labels are used for training the classifier and this is called supervised learning. After the classifier has been trained, it can predict which class or in our case, visual expression, new data belongs. Popular classifiers in facial expression recognition are the Support Vector Machine [11-15], Bayesian Networks[16-18] and Neural Networks[19-22]. Specifically, SVM is reviewed and applied as it usually achieves decent performance among types of classifiers and is most commonly used in this area. This makes it a suitable method to study and be used for comparison.

2.1.3.1 Support Vector Machine

Support vector machine[70, 71] creates a hyperplane in a high dimensional space (for binary classification). This hyperplane divides the two classes in terms of making it the largest distance to the nearest training points of both classes. So the classifier could generate less classification error as the margin is larger.

Assume the training data *D* has n points:

$$D = \{(\boldsymbol{x_i}, y_i) | \boldsymbol{x_i} \in \mathbb{R}^p, y_i \in \{-1,1\}\}_{i=1}^{n} \qquad (2.11)$$

Where $\boldsymbol{x_i}$ is training point as a p-dimensional real vector and y_i is the corresponding class with the value of -1 or 1. Any hyperplane can be written as

$$\boldsymbol{w} \cdot \boldsymbol{x} - b = 0 \qquad (2.12)$$

Where **w** is denotes the normal vector to the hyperplane. The $\frac{b}{\|\boldsymbol{w}\|}$ determines the shift from the origin. Then the correct classification should satisfy the equations

$$\boldsymbol{w} \cdot \boldsymbol{x_i} - b \geq 1 \qquad \text{for } \mathbf{x_i} \text{ of the first class} \qquad (2.13)$$

$$\boldsymbol{w} \cdot \boldsymbol{x_i} - b \leq -1 \qquad \text{for } \mathbf{x_i} \text{ of the second} \qquad (2.14)$$

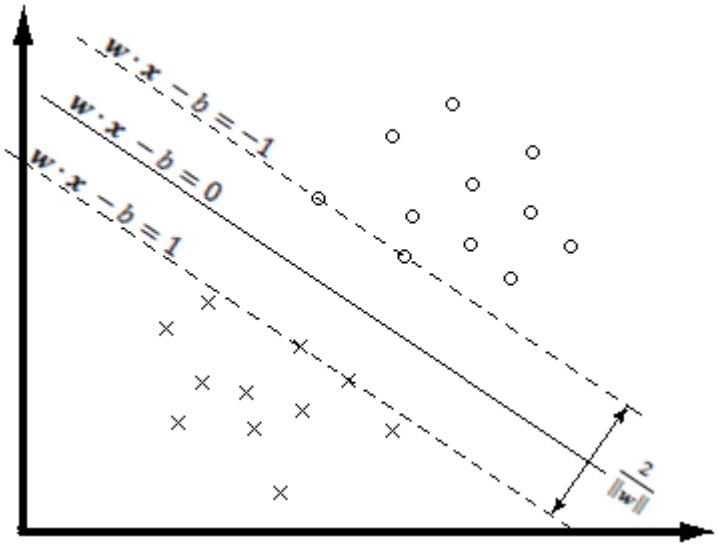


Figure 2.8: Margin for SVM trained on two class samples

The distance between two classes, as shown in figure 2.8, is $\frac{2}{\|\boldsymbol{w}\|}$. So minimise the $\|\boldsymbol{w}\|$ to have

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} - b) \geq 1 \qquad \text{for any i=1,2,…,n} \quad . \qquad (2.15)$$

Substitute $\|\boldsymbol{w}\|$ with $\frac{1}{2}\|\boldsymbol{w}\|^2$ to make the computation easier without changing the solution. Then minimising $\frac{1}{2}\|\boldsymbol{w}\|^2$

$$\min_{(w,b)} \frac{1}{2}\|\boldsymbol{w}\|^2 \qquad (2.16)$$

with the constraint of equation 2.15 is a quadratic programming optimisation problem. By introducing Lagrange multiplier α, the constrained problem can be described as

$$L(\boldsymbol{w}, b, \alpha) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \alpha_i (y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) - 1) \qquad (2.17)$$

The "stationary" Karush–Kuhn–Tucker condition implies that the solution can be expressed as a linear combination of the training vectors

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x_i} \qquad (2.18)$$

Using the fact $\|\boldsymbol{w}\|^2 = \boldsymbol{w} \cdot \boldsymbol{w}$ and equation 2.18, the Lagrange duel is

$$L(\boldsymbol{w}, b, a) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x_i^T x_j} \qquad (2.19)$$

Subject to (for any i =1,2,…,n)

$$\alpha_i \geq 0 \qquad (2.20)$$

And to the constraint form the minimization in b

$$\sum_{i=1}^{n} \alpha_i y_i = 0. \quad (2.21)$$

Now we can calculate $\boldsymbol{w}$ after obtaining α term based on equation 2.21.

In equation 2.19, $\boldsymbol{x_i^T x_j}$ can be replaced by a kernel function with the same dot products

$$\boldsymbol{x_i} \cdot \boldsymbol{x_j} = k(\boldsymbol{x_i}, \boldsymbol{x_j}). \quad (2.22)$$

With the kernel function, the data can be mapped into a higher dimension space. Then a suitable kernel function may help to solve a non-separable classification on the old dimension space. Some common used kernel functions are

Polynomial (homogeneous): $k(\boldsymbol{x_i}, \boldsymbol{x_j}) = (\boldsymbol{x_i} \cdot \boldsymbol{x_j})^d$ (2.23)

Polynomial (inhomogeneous): $k(\boldsymbol{x_i}, \boldsymbol{x_j}) = (\boldsymbol{x_i} \cdot \boldsymbol{x_j} + \mathbf{1})^d$ (2.24)

Radial Basis Function (RBF): $k(\boldsymbol{x_i}, \boldsymbol{x_j}) = \exp(^{-\|\boldsymbol{x_i} - \boldsymbol{x_j}\|^2}/_{2\sigma^2})$. (2.25)

The RBF kernel is commonly used the in SVM.

2.2 Evolutionary Algorithms

Evolutionary Algorithms are inspired by Darwin's theory of evolution[72]. Darwin suggested species survived over time because they could adapt to the environment. The individuals that are fitter to the environment have a higher chance of survival, and only those individuals have the chance to produce offspring that carry on the genes to future generations. This is termed *natural selection*. It is difficult for the individuals with less fit genes to survive and so there is less chance the next generation will inherent these less fit genes, whilst the genes which enable creatures to survive will, over time, become more prevalent in the entire population.

We may ask ourselves, from where do the good genes and bad genes originate? They are a combination of the parents' genes but can also result from random changes or *mutation*. There is no influence on the mutation of gene and so the fitness of the resulting individual can be either good or bad. However, as stated above, natural selection favours fitter genes and so only those individuals with fitter genes will be selected and inhereted, causing evolution to take a certain direction towards better adaptation to the environment. The speed of evolution is typically slow and depends on the speed and quantity of reproduction and, hence, species may die out when the environment changes so suddenly that there is no time for evolution.

Evolutionary Algorithms (EAs) are a generic population-based metaheuristic optimization algorithm[23]. In EAs, one individual represents a possible solution to a problem. There are a number of solutions in the population pool and a fitness function is used to evaluate how well these solutions can solve the problem at hand. Individuals with higher fitness are selected and used to produce offspring which are expected inherent high fitness. The offspring are produced by genetic operations such as crossover and mutation and will normally result in individuals with different fitness values. By repeating this selection and reproduction operation for each generation, the average fitness of population should gradually increase, and hence, the ability to solve the problem will improve. After a number of generations (that can be determined by a number of factors) a solution is obtained which is the fittest so far of those evolved for the problem under consideration. A more detailed description of the structure and functioning of evolutionary algorithms follows in the next section.

2.2.1 General Structure of EAs

This section will review the functioning of EAs in more detail and will consider the Genetic Algorithm (GA) as example. The GA is probably the most widely used and well known EA and has been well studied over the past decades. It is used for optimisation as well as solving general search problems.

The solution expected from a GA comprises numerical parameters which are used to configure some other systems to have better performance. These numbers are usually converted to binary form in a fixed length string, which can be considered to be one individual.

2.2.1.1 Initialisation

At the start of the process, a population of solution individuals are generated, each individual typically a binary string is initialised randomly. The values assigned to these individuals are not significant but should be within a predetermined range, if appropriate to the problem under consideration. The generation of individuals continues until the required population has been achieved. The population size depends on the problem and processing resources available. A larger population size requires more computational resources but may achieve a satisfactory result in fewer generations, and vice versa.

2.2.1.2 Evaluation

Evaluation of an individual's fitness requires the binary string to be evaluated in the following way. The individuals are converted back to numeric parameters, and then applied back to the original problem to calculate the error between the output obtained and expected result. This error can be used as the fitness value that represents how well the individual has performed. The way to evaluate the fitness value can vary, but is usually a simple and direct method to minimise computational effort. The function to evaluate the performance of an individual is called the *fitness function*.

2.2.1.3 Selection

After all the individuals in the population have had their fitness values evaluated, a number are selected to produce offspring, or individuals, for the next generation. A common selection

method favours individuals with higher fitness, whilst less fit individuals have a lower chance of being selected. To preserve diversity, it is better not to choose only the fittest individuals for reproduction as they may be genetically very similar, resulting in the search terminating prematurely on a local optimal.

2.2.1.4 Crossover and Mutation

This step is going to generate next generation from selected individuals. Individuals selected from last step are called. A pair of parents produces child solution by crossover and/or mutation. Cross over is a recombination of two parent solutions. Mutation is a random position of the binary string from the solution becomes another random number in a certain chance. Examples of crossover can be found in the figure 2.8 – 2.12. An example of mutation is also given in figure 2.12. The child solution is different from parents but comes from them and is considered inherent their characteristics. The child solutions will keep being created until the population pool is filled up with the new generation.

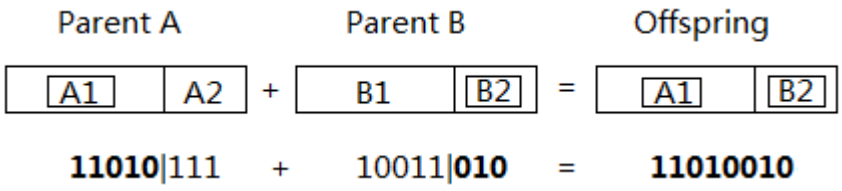


Figure 2.9: Single point crossover

Single point crossover is to choose one point in the chromosome. It divides the chromosome into two pieces. The offspring come from one piece of parent each.

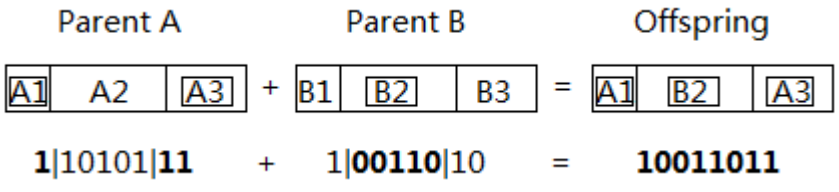


Figure 2.10: Two points crossover

In figure 2.9 two points are selected and the chromosome is divided into three pieces. The first and last pieces are from one parent and the middle piece is from the other parent to form the new offspring.

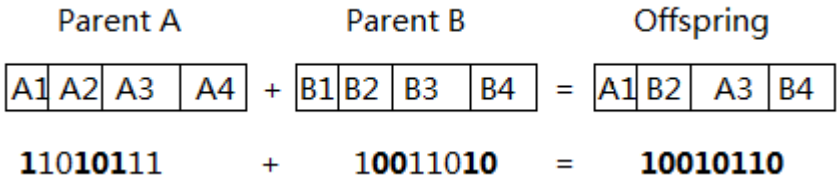


Figure 2.11: Uniform crossover

In uniform crossover, the bit on the chromosome is chosen from one of parents randomly at the same chromosome position.

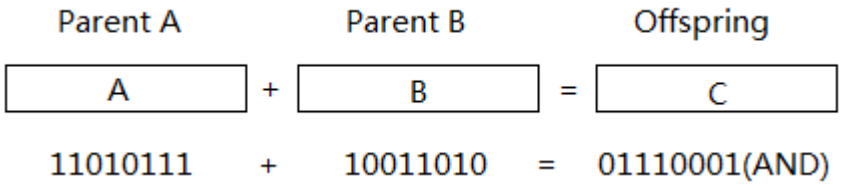


Figure 2.12: Arithmetic crossover

In figure 2.11, the offspring does not copy the chromosome from parents directly but a new from arithmetic operation of parents' chromosomes.

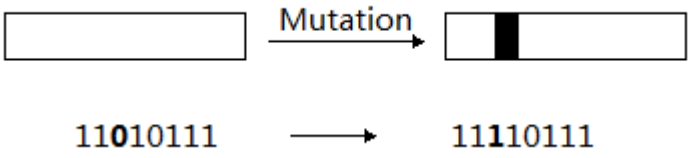


Figure 2.13: Example of mutation

Mutation is to invert a random gene on chromosome as shown in figure 2.12.

2.2.1.5 Repeat and Terminating Condition

The selection and reproduction process is repeated for every new generation. After many generations, the average fitness should increase, suggesting the evolved solutions will perform better. The process will stop when the *terminating condition is reached*. Terminating conditions can include: a satisfactory solution has found; no better solution can be found; the fixed number

of generations has reached. The individual with the best fitness is the solution obtained from one run of the GA. From the terminating condition, the solution is either the one that meets the requirement or the best one under a certain processing time.

2.2.2 Types of Evolutionary Algorithms

Evolutionary algorithms include Genetic Programming (GP), Evolutionary Programming (EP), Evolutionary Strategies (ES) and Genetic Algorithms (GAs). They all have a similar structure of evolution, but are applied to different applications due to variance in functionality and structure. GP and EP are both algorithms that generate solutions that represent a computer program. The fitness value is evaluated by how well the solution can solve a computational problem. The difference is that in EP only the numerical parameters can be evolved but the structure of the program is fixed, in GP both can change during evolution. GAs are the most popular and well known EAs and are often used for solving optimisation problems. ES is often used as the underlying evolution process of the above algorithms.

2.2.3 Genetic Programming

A computer is expected to follow explicit commands, rather than how to solve the problem and, consequently, computers still require human programming achieve this. Alan Turing was perhaps one of the first to provide an alternative point of view: “a computing machine could manipulate itself just as well as any other data”[73]. GP is a form of self-manipulating computing machine, as a member of EAs.

2.2.3.1 What Can GP Do?

GP has attracted a considerable researcher interest through its ability to automatically evolve programs. It has been applied to many fields, not only in assisting software programming and Artificial Intelligence (AI) game design, it has also been applied to areas such as image processing, classification, function optimisation, financial market prediction and medical diagnosis[74-92].

Programming and Gaming Artificial Intelligence (AI): Westley Weimer et al. proposed a method that can automatically locate and repair bugs in software by GP[74]. It successfully repaired some 6300 lines of code in ten different C programs within an average time of 200 seconds. GP is also suitable for developing AI. Yehonatan Shichel et al. developed AI by GP for the *Robocode* game

and the resulting bot was the only one entered in the tournament that not written by a human[75]. Ivan Tanev used GP for designing the locomotion gaits of a snake-like robot automatically[76]. There are also other work on GP based gaming - AI. Ami Hauptman and Moshe Sipper designed AI by GP for chess endgame[77]; Yaniv Azaria and Moshe Sipper used GP to evolve AI for the game backgammon[78]; David Jackson proposed a method that evolves defence strategies of games by GP[79].

Image Processing and Machine Learning: Riccardo Poli proposed that GP had potential in image processing. He considered image enhancement, feature detection and image segmentation as filtering problem. Then GP was applied for discovering efficient optimising filters. He applied this method to brain image segmentation for medical use[80]. Jay F. Winkeler and B. S. Manjunath tried to use GP for detecting faces in images[81]. Jamie R. Sherrah et al. used GP for automatic feature extraction and tested it using different classifiers[82]. They found a phenomenon that a classifier with malleable decision boundaries will impede evolution of GP for feature extraction. GP had better optimisation performance when combined with a simple classifier. L. Guo et al. used GP to discriminate characters in damaged documents[83]. Aydın Akyol et al. applied GP to classify pollen cell images[84]. Hong Guo et al.[85], and Durga Prasad Muni et al.[86] also used GP for feature extraction in separate projects.

Machine learning is another important application of GP. GP has been used as classifier for many fields. Thomas Mckee and Terje Lensberg proposed a GP model for bankruptcy prediction[87]. They achieved 80.3% classification accuracy which is significantly better than 67% classification accuracy achieved using the rough sets model. Liang Zhang and Asoke K. Nandi applied a GP classifier to roller bearing fault detection[88]. GP classifier can be also used for medical diagnose and disease prediction. Stephan Winkler et al. proposed enhanced GP classifier, which combined logical expressions and classical mathematical functions. They trained the classifier with five data sets that comprised medical measurements of patients potentially suffering skin cancer[89]. R. J. Nandi et al. applied GP to classify breast masses located in mammograms that indicate cancer[90]. Hong Guo and Asoke Nandi also proposed a method that used features generated by GP to diagnose breast cancer[85]. Jin-Hyuk Hong and Sung-Bae Cho applied GP for the classification of cancer based on DNA microarray data[91]. Topon Kumar Paul and Hitoshi Iba used GP for cancer prediction based on gene expression data[92].

2.2.3.2 GP Algorithm Discussions

Chris Gathercole and Peter Ross discussed population size of GP in their paper[93]. They suggested a small population size over more generations was better than a large population size

over few generations, both in terms of speed and performance. The diversity of a population is important for GP because it is crucial to avoid premature convergence toward local optima. Edmund Burke et al. tried to find the correlation between diversity and best fitness of population in GP[94]. Code growth is one interesting feature of the program generated by GP. The programs invariably grow in size including a large number of non-functional codes which do not affect the program. Terence Soule et al. described this phenomenon and suggested adding a penalty for large solutions in the fitness function was a possible way to limit this behaviour. Markus Brameier and Wolfgang Banzhaf proposed a new variant linear GP and deleted such non-functional codes that did not affect program behaviour[95]. By doing this, the length of the program was shortened and the speed was increased but did not change the result. W. B. Langdon and R. Poli under took similar work by analysing the MAX problem in GP[96].

Sean Luke and Lee Spector compared crossover and mutation in GP. They finally found the crossover was slightly better than mutation (primarily larger population size), but the difference was surprisingly small[97]. William Punch considered the behaviour of multiple populations for parallel GP[98].

2.2.3.3 GP Classifier

The major application of GP is in machine learning and the classification problem is the most prolific of all applications. It has been studied over several decades and a large number of GP classifier techniques have been developed. Current GP classifiers not only vary in types of structure, but also have many variations in any step that try to solve any problems met during the classification process.

Structure

A parse tree is the basic structure of standard GP invented by Koza[99]. It is a tree-like structure which executes from leaves to root. The leaf nodes are the terminal arguments for the functions represented as internal tree nodes. Each function gets arguments from its child nodes. Follow this path and the output of program can be found at the root. One reason of choosing parse tree is the programming language LISP can be represented by parse tree by its natural structure. LISP was exactly the language Koza first used for GP. Then it is possible to have the class label by thresholding the output.

Cellular GP has the structure of multiple layers[100]. The program represented by the tree of the first layer creates the structure of the tree in the next layer. The whole structure is not evaluated directly, but by stages, from one layer to the next. Each layer has its own input and output, the

output of one layer providing the input to the next layer. Linear GP is another important type of GP. It is represented by a list of machine language instructions.

Celia Bojarczuka et al. applied a new constrained-syntax (GP) algorithm for discovering classification rules [101]. Jung-Yi Lin et al. introduced a novel method called *FLGP* for multi-population GP[102]. It had the multi-layer structure that one layer outputs features to the next layer as its input. Durga Prasad Muni et al. proposed a novel method that provided more opportunity to turn "unfit" trees to fit[103]. Also they introduced "OR-ing" chromosomes which made the classifier perform better. Athanasios Tsakonas compared the performances between four GP based classifiers including: decision trees, fuzzy rule-based systems, neural network (NN) and fuzzy Petri-nets with GP[104]. Waranyu Wongseree et al. compared the Thalassaemia classification performance between GP and NN[105]. Decision tree based GP with nonlinear fitness function had the similar performance as one hidden layer NN. Two hidden layers NN had better performance.

F. J. Berlanga introduced a fuzzy rule-based system with GP-COACH, and it performed better than other well-known fuzzy rule-based classification systems[106]. Roberto R.F. Mendes et al. proposed a system using GP that evolved a population of fuzzy rule sets and a simple evolutionary algorithm evolving a population of membership function definitions[107]. The two populations co-evolved, so that the final result of the co-evolutionary process was a fuzzy rule set and a set of membership function definitions which were well adapted to each other.

Function set

The functions used in GP generally depend on the actual problem to solve. In the numeric problem, the elementary algebra is sufficient for the function set. But basically all kinds of functions and constructs can be used, not only logical operations like 'and', 'or' and 'not', conditional 'if', 'then', but also filters for image processing tasks, computer commands for program generating.

Table 2.1: Examples of primitives in GP function and terminal sets[23]

Function Set	
Kind of Primitive	Example(s)
Arithmetic	+, -, ×, /
Mathematical	sin, cos, exp
Boolean	AND, OR, NOT
Conditional	IF-THEN-ELSE
Looping	FOR, REPEAT
…	…

Fitness Function

The fitness function is used to evaluate the performance of the algorithm on one aspect or measure in EAs, but many problems have more than one requirement which may be in conflict. In such *multi-objective* optimisation problems, it is usually not possible to obtain a single solution that has superior performance over all requirements, but a set of solutions that are *nondominated*, where there is no one solution that is better than another. In this case, any single solution cannot improve the performance of one requirement without decreasing the performances of another. This problem is also known as the *Pareto front[108]*. Kalyanmoy Deb[108], Daniel Parrott et al.[109], Mengjie Zhang and Urvesh Bhowan[110], Peter Lichodzijewski and Malcolm Heywood[111] have all published extensive work in this area.

Output

For a binary classification problem, a simple threshold function can be used to decide the assigned class from a real value. But for a multi-class classification problem, five methods are summarised by Thomas Loveard and Victor Ciesielski[112]. These methods are: binary decomposition, in which the problem is decomposed into a set of binary problems and standard genetic programming methods are applied; static range selection, where the set of real values returned by a genetic program is divided into class boundaries using arbitrarily chosen division points; dynamic range selection in which a subset of training samples are used to determine where, over the set of reals, class boundaries lie; class enumeration which constructs programs similar in syntactic structure to a decision tree; and, evidence accumulation which allows separate branches of the program to add to the certainty of any given class. In further examples, Chi Zhou et al. treated the multi-class classification task as a multiple binary classification problem[113], whereas Mengjie Zhang and Will Smart used dynamic boundaries to discriminate classes instead of predefined static thresholds[114].

Training Strategy

The imbalanced data classification problem, is a situation in which the amount of training data available for one class is significantly more (or less) than that of another class[115]. It usually results in a poor accuracy for minor class samples. To overcome this problem, the training data can be reorganised to be balanced by selecting a subset of the major class's data or by interleaving extrapolated samples within the minor class's data. Grant Patterson and Mengjie Zhang tried to modify the fitness function so that the minor class data and major class data were evaluated with comparable weightings instead of using a standard overall average recognition rate. This method increased the accuracy of the minor class but the rate of major class dropped considerably[116].

J. K. Kishore et al. proposed an incremental learning method where the GP evolves part of the population for certain generations, and then increase the number of population for training for another number of generations[117].

Over fitting occurs when a statistical model describes random error or noise instead of the underlying relationship[118]. The same concept can apply to evolutionary algorithms where the solution also describes random error or noise in the data[119]. Matthew Smith and Larry Bull indicated that randomly reordering the training data during the process may help to reduce the effect of over fitting[120].

2.2.3.4 Cartesian Genetic Programming

Cartesian Genetic Programming (CGP) is a highly efficient and flexible form of Genetic Programming that represents a computer program as a graph matrix structure. It was invented by Julian Miller in 1999 and was developed from a representation of electronic circuits devised by Julian Miller and Peter Thomson[121]. CGP represents computational structures as a string of integers. These integers, known as genes determine the functions of nodes in the graph, the connections between nodes, the connections to inputs and the locations in the graph where outputs are taken from.

CGP has been applied to many different areas and applications, including machine learning[122], neural networks[123], artificial intelligence[124], financial prediction[125], function optimization[126], classification[127], electronic circuit design[128, 129], medical diagnostics[130], evolutionary art[131] and music. The detail of CGP will be introduced in chapter 4.

2.3 Summary

In this chapter, the background of expression recognition and evolutionary algorithms is introduced and related literatures are reviewed.

The process of the facial expression recognition normally includes three steps: face acquisition, feature extraction and classification. Viola and Jones' algorithm is the most successful means of real time face detection. Feature extraction is usually essential for the system to reduce the dimensionality of data by transformation. The data of whole face area is suspected to be redundant and the discriminating information is expected to remain after the feature extraction. The common used feature extraction methods for recognising expressions includes: Gabor wavelet, Local Binary Pattern, Active Appearance Model and Optical Flow. The former two are mainly for static image analysis. The advantage of AAM is that it can detect human face and fiducial points at the same time. Then the tracking of fiducial points can be good representation of dynamic face information for expression recognition. Optical Flow estimates the displacement of each pixel in the image between frames so it is suitable for dynamic analysis. It can be used for tracking the fiducial points (sparse optical flow) extracted by other algorithms or extracting dense optical flows of all pixels on the face.

The classification is the final step for telling the expression name based on the features input after supervised training process. Currently the classifiers used for expression recognition are mainly conventional or statistics classifiers, like Support Vector Machine, Bayesian Networks. There are also a number of methods used Neural Networks for classification. But not much research has been done on expression recognition by Evolutionary Algorithms.

Evolutionary Algorithms follow the idea of Darwin's theory of evolution. In the solution population the fitness values of individuals are evaluated based on the performance. The solutions have better performance will survive and produce offspring by crossover and/or mutation. Then the next generation is supposed to inherent high fitness from the parents. After many generations, the average fitness in the population is increasing. At last the best solution (with highest fitness value) is the final result. Genetic Programming is one of EAs for generating programs or computations automatically. Cartesian Genetic Programming is a flexible form of GP and it can be used for classification problem as well. Details of CGP will be introduced in the beginning of Chapter 4.

3 Optical Flow for Feature Extraction

3.1 Introduction

A large proportion of expression recognition research works with the static image[12, 33, 50-52]. Individual frames taken from a video stream or image sequence is analysed to give a prediction of the facial expression. This kind of expression recognition technique is based on the appearance at a single time point, but appearances may vary greatly between different people, even for the same expression. Also, it may be not reliable or practical to detect every frame separately.

The transition of an expression, from neutral to a smile, for example, potentially contains much useful information and may be more reliable than just the expression itself. Recent research has turned from considering just static image analysis to the dynamic analysis of expressions. Ekman's research indicates that muscles activated for one expression are the same for different people[132]. Therefore, the changing of appearance caused by these muscles could be very useful for expression recognition. An effective way to extract the transition of these muscles is to use optical flow, which estimates motion between two images or frames of a video. In this chapter, a dynamic expression recognition method is proposed based on optical flow and investigated how to apply to sample videos.

A database of suitable facial expressions is essential for undertaking this research, so identifying a suitable database is the first important task to solve. The requirement for a database set out for this research is a dynamic expression dataset from video or image sequences that has six universal expressions (happiness, anger, surprise, sadness, fear and disgust), which are naturally expressed and not acted. The creation of a custom database is also considered in this chapter.

As previously discussed, dynamic analysis for expression detection has advantages over static image analysis. Optical flow is the technique chosen to estimate the motion between two frames of video which is suitable for extracting useful information in expression transition. Currently, most dynamic methods apply *sparse optical flow* on some *fiducials* (or reference points) within the image[9]. This type of method locates the most important points on the face, usually the eyes, corners of the eyes, eyebrows and mouth corners. It then tracks these points between frames using the optical flow algorithm. The movements of these points are taken and used for detecting facial expression, but these methods also have their limitations:

1. The model and method for extracting fiducial points can cause additional errors. Although the same muscles used for one expression are the same for most individuals, the exact location of the muscles and appearances generated may differ. Therefore, additional algorithms are required

for locating these points and errors may confuse the true motion. This can be crucial because the differences in muscle movement between two expressions can be small.

2. Sparse optical flow may lose important information. Although taking a limited number of points on the face will shorten the processing time, which can be important for real-time classification, loss of information can lead to greater error. The changing appearance due to muscles contracting or expanding relies on more than just these points, but they are often used because they are easy to locate as high contrast or corner points. Certainly, from a human perspective, recognising facial expressions involves more complex processing.

To address these problems an alternative, *dense optical flow* was used. However, a problem associated with this technique is that the resulting vector length is too large, which makes the classification difficult, as much of the optical flow detected on the face contains redundant data. Therefore, additional feature selection or dimensionality reduction needs to be applied to dense optical flow data to make it usable. In this chapter, implementations of three such approaches are considered, the most effective of which is subsequently used for the comparison of classifier performance.

3.2 Databases

In order to study the human facial expression, a database of samples is essential. As this research is concerned with dynamic analysis of expression recognition, video or image sequence databases were considered instead of static image databases. A database should be constructed to accommodate as many conditions as possible to make it reliable and authentic. Users can also select the database that is most suitable for their research by studying these conditions and other characteristics. The research considered here requires video data of six universal expressions, preferably naturally expressed rather than intentionally exaggerated or acted. The MMI database and FG-NET databases are now considered along with a custom-developed database.

3.2.1 MMI Database

The MMI expression database was conceived in 2002 by Maja Pantic, Michel Valstar and Ioannis Patras[133]. It was created as a resource for building and evaluating facial expression recognition algorithms. Compared to other databases, it is easy to access through a web site, and convenient to search in terms of expression, gender and age, for example. It contains both static images and videos of expressions. Other properties are summarized below.

Face Pose

Most research focuses on the front face analysis, because this contains the most information, but researchers are also working on other viewing angles to make the analysis more reliable. The video sequences in this database have two types of camera angle: frontal and profile-view. There are approximately 750 dual-view facial expression video clips.

Candidates

The MMI database has a great variety of candidates although in the beginning they are university students and staff members. Some 44% of them are female and the ages of candidates range from 19 to 62. They have either a European, Asian, or South American ethnic background.

Expressions

The database includes six universal and some other additional expressions. Initially, candidates were asked to intentionally produce an expression - later natural expressions were also added. For the intentional expressions, the video captures the complete transition from neutral to an expression, and then back to neutral. There are 39 video clips for happy, 28 for angry, 36 for surprise, 23 for fear, 19 for sad and 32 for disgust, from MMI used in this thesis.

Labels

Some data has been coded using the Facial Action Coding System (FACS)[31]. This can be the ground truth for training or testing of recognition system. It makes the user easier otherwise they have to mark Face Action Units themselves if necessary.

3.2.2 FG-NET Database

The FG-NET database[134] was constructed for collecting the natural expressions of real emotions. It records the entire head, while playing video examples to arouse a specific emotion. The candidates are asked not to act the expression and are free to move their head. With these strategies, they aim to generate an expression database that is as natural as possible.

Face Pose

The frontal face is recorded in this database, but because the candidates may move their head during recording, the camera frames the entre head in the picture instead of just the face.

Expressions

The database has six universal expressions plus a neutral expression. Each expression is represented by three video clips for each candidate. There are 18 candidates in total and 324 video clips used from FG-NET, exclude neutral expression, in this thesis.

Labels

The first frame of every video clip was labelled as the reference frame. The labels include the centre coordinates of the eyes, nose and the mouth.

3.2.3 Custom-developed Database

A custom-developed database was also generated. The aim was to obtain natural and acted expressions from the same person, to explore whether it is possible to automatically detect the difference between real and fake expressions.

Recording Conditions

To achieve high quality videos of facial expression, the following conditions were set for the recording: a constant and simple background, frontal face angle and good illumination. No other people were permitted in the room who might interfere with the candidate.

Videos for Evoking Emotions

The videos presented to candidates for evoking emotions should be simple and direct, to ensure candidates will exhibit the desired corresponding expression. Sadness is difficult to arouse, so the video needs to be longer to evoke this expression. However, a long video makes the transition of the sad expression longer as well, which makes it difficult to extract the optical flow data. Another difficult expression to motivate is anger. The corresponding video is difficult to achieve that satisfies the effect without causing undue offence. In total, 3 clips for happiness, 2 for scary, anger, surprise, fear, sadness and disgust are chosen for the generation of database.

The clips are compiled in one video in a random order, but the happy videos follow sad and disgusting videos to avoid making the candidate feel uncomfortable. There is a five second break between each clip, during which the candidates expression is expected to relax from the previous emotion. After all the video clips have been presented, the candidates are asked to act out the expressions intentionally. Each expression is asked of the candidates twice and two different images with that expression are shown on the screen separately to guide the tester in case they do not know how to act that expression. The length of the whole video is made to be less than 15

minutes, because too much time will tire the candidates. The process consists of playing the whole video and recording the complete reaction of the candidate. In total 37 candidates took part, resulting in 37 videos of approximately 15 minutes length each. The candidates are mostly the university students in China.

The resulting database is expected to use for studying the facial expressions and analyse the difference between real expressions and fake ones from same person. From the feedback of the candidates, the sad and disgust videos made some of them feel uncomfortable, so the videos with negative emotions should be carefully chosen. If the video is too strong, it will make the candidates feel uncomfortable, but if too weak, it will not evoke a satisfactory response from the candidates; this was particularly an issue experienced in the anger emotion recording.

3.2.4 Databases Used in the Project

Both the MMI and FG-Net databases are used in this project. Six universal expressions are used for training and testing. The video in the database was cropped around face area as this is the only part of the whole video which is required for these experiments. At the same time, the face size and position within the picture are controlled to be similar to the results of detection window by Viola and Jones' algorithm. So the classifiers trained with the data can be used for the prediction on a new face from detection function later. Only the rising part of each expression is used for feature extraction. Time tags for the start of the expression and reaching the maximum or full expression are added to a file that contains the video name. These are used for processing the data at a later stage.

Our own database is not taken for the experiments. The main reason is the expressions recorded are not obvious enough from the observer's view. As not too much changing on face, the data is not suitable for this optical flow based experiment. Some candidates do not react to the evoking video at all. There are three possible reasons: they have watched the videos before; they have no sense about the video; they were evoked the emotion but not strong to show on face. Although the expressions evoked by our videos are not strong, they can be considered natural and may have other use in future. The second part is to let the candidates act expressions with hint of an example image shown on screen. The recorded expressions are not representative either. It is difficult to make a general people act an expression without training and a guide image is not enough. Some of candidates do not act at all, some others try to perform but the appearances do not make people understand the expected emotion.

Actually, for an expression database, the ideal expressions are clear and representative, and for expression recognition, a plain face for training and testing does not help too much. Strategies

have been used in other existing databases including: teach the candidates which muscles to use to perform an obvious and 'correct' expression; hire professional actors to perform the expressions; encourage the candidates act as exaggerate as possible, then give the expressions marks by a group, the data with high scores will be used. These methods are help to obtain the videos with obvious expressions and letting observer know what expression it is. The drawback is they are not natural expressions. The best way possibly is to create brand new and high quality evoking videos; this may need psychologists' help. Make sure the candidates fully cooperate and in a relax condition. The recording time should be as short as possible, this also require the evoking video can make people quick response.

It was, however, invaluable for understanding the limitations of evoking natural expressions compared to acted expressions. The experiences on how to create an expression video database has been obtained, it also helps to understand and have good use of other databases. The video data is not useless and it can be used for other suitable experiments in the future.

3.3 Optical Flow

Optical flow was introduced in Chapter 2, but an additional constraint is required to solve the *aperture problem*. Amongst the most common constraints applied are Horn and Schunck's smoothness constraint for dense optical flow[62] and Lucas and Kanade's constraint for sparse optical flow[63]. These methods, which are described in the following sections, are required for fast and accurate estimation of optical flow.

3.3.1 Horn Schunck Optical Flow

3.3.1.1 Smoothness Constraint

If each point moves independently, it is not possible to estimate the velocities because of the lack of information to solve the equations. Horn and Schunck[62] introduced a method by using the smoothness constraint to address this issue. The movement of an object in vision is the motion or deformation of a certain area of points. Most of the time, a group of points' movements are rigid, so the velocities of neighbouring points on the object are similar and, hence, the velocity field of points varies smoothly throughout the whole image. There is an exception - if one object blocks another, the flow will discontinue at the connection of objects. In this case, the smoothness constraint is unlikely to perform well.

The constraint can be expressed as minimising the square of the magnitude of the gradient of the optical flow velocity:

$$\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 \qquad (3.1)$$

and

$$\left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \qquad (3.2)$$

Where "u" means the velocity on the x axis, "v" means the component of flow on the y axis.

The Laplace operator is another expression used to measure the smoothness of the optical flow field:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \qquad (3.3)$$

and

$$\nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \qquad (3.4)$$

So the total error to be minimised is:

$$E(u,v) = \iint \left[(I_x u + I_y v + I_t)^2 + a(\nabla^2 u + \nabla^2 v) \right] dxdy \quad (3.5)$$

where $(I_x u + I_y v + I_t)^2$ is the error of the change of image brightness. I_x is the partial derivative of image brightness with respect to x. It is the same situation with the other two: I_y of y, I_t of t. Expression $\nabla^2 u + \nabla^2 v$ measures the smoothness of optical flow field where "a" is the weight of the two factors.

By achieving the minimisation of equation 3.5, the optical flow velocity is the corresponding value of *u* and *v*. By applying the calculus of variation equation 3.5 can be transformed as follows:

$$I_x^2 u + I_x I_y v + I_x I_z - a\nabla^2 u = 0 \qquad (3.6)$$

$$I_x I_y u + I_y^2 v + I_y I_z - a\nabla^2 v = 0 \qquad (3.7)$$

The way to estimate I_x, I_y and I_z can be found in section 3.3.1.2, the approximation of *u* and *v* is described in section 3.3.1.3.

3.3.1.2 Estimating the Partial Derivatives

The derivatives of brightness can be measured from the images. The value of $I_x I_y I_z$ should be estimated consistently by referring to the same point or pixel in the image each time. One method of estimation is by forming a cube specified by eight measurements. The structure of the cube and relationship of the measurements are shown in figure 3.1.

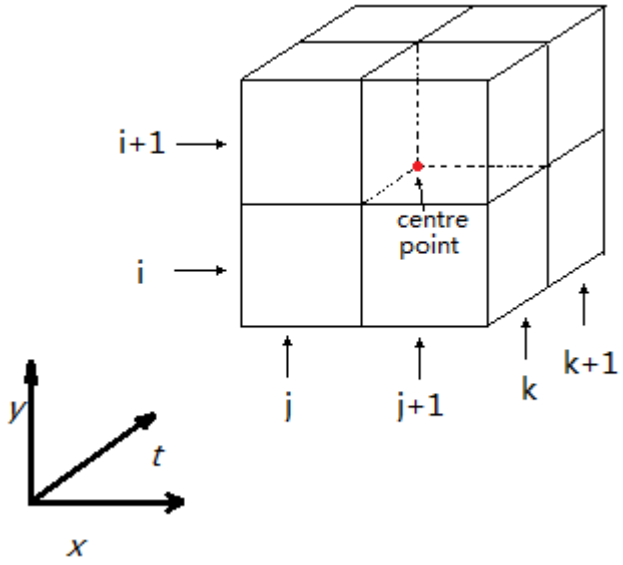


Figure 3.1: Cube structure adopted in the estimation of the derivatives of brightness

The red point in the centre of the cube is the point at which the partial derivatives are to be estimated. Each derivative is estimated from the average of four parallel groups of differences between two adjacent measurements, where x and y are axes of the image, t axis is the frame of image sequence, means time. Indexes j, i, k are in x, y, t axes respectively.

Then the partial derivatives of one point in the image can be estimated by the following equations:

$$I_x \approx \frac{1}{4}\left[\left(I_{i,j+1,k} - I_{i,j,k}\right) + \left(I_{i+1,j+1,k} - I_{i+1,j,k}\right) + \left(I_{i,j+1,k+1} - I_{i,j,k+1}\right) + \left(I_{i+1,j+1,k+1} - I_{i+1,j,k+1}\right)\right] \quad (3.8)$$

$$I_y \approx \frac{1}{4}\left[\left(I_{i+1,j,k} - I_{i,j,k}\right) + \left(I_{i+1,j+1,k} - I_{i,j+1,k}\right) + \left(I_{i+1,j,k+1} - I_{i,j,k+1}\right) + \left(I_{i+1,j+1,k+1} - I_{i,j+1,k+1}\right)\right] \quad (3.9)$$

$$I_t \approx \frac{1}{4}\left[\left(I_{i,j,k+1} - I_{i,j,k}\right) + \left(I_{i+1,j,k+1} - I_{i+1,j,k}\right) + \left(I_{i,j+1,k+1} - I_{i,j+1,k}\right) + \left(I_{i+1,j+1,k+1} - I_{i+1,j+1,k}\right)\right] \quad (3.10)$$

3.3.1.3 Estimating the Laplacian of the Flow Velocities

The Laplacian of u and v can be estimated by using the discrete Laplacian operator[62], which is often used in image processing in edge detection and motion estimation applications. The discrete Laplacian is calculated as the sum of differences over the nearest neighbours of the central pixel. The weights of those neighbours are illustrated in figure 3.2.

1/12	1/6	1/12
1/6	-1	1/6
1/12	1/6	1/12

(a)

The weights used in Horn and Schunk's paper[62]

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

(b) (c)

Other commonly used weights of Laplacian

Figure 3.2: Examples of Laplacian weights

Therefore, the Laplacians of u and v can be estimated based on the weights shown in figure 3.2 (a):

$$\nabla^2 u \approx \frac{1}{6}\left(u_{i-1,j,k} + u_{i,j+1,k} + u_{i+1,j,k} + u_{i,j-1,k}\right) + \frac{1}{12}\left(u_{i-1,j-1,k} + u_{i-1,j+1,k} + u_{i+1,j+1,k} + u_{i+1,j-1,k}\right) - u_{i,j,k} \qquad (3.11)$$

$$\nabla^2 v \approx \frac{1}{6}\left(v_{i-1,j,k} + v_{i,j+1,k} + v_{i+1,j,k} + v_{i,j-1,k}\right) + \frac{1}{12}\left(v_{i-1,j-1,k} + v_{i-1,j+1,k} + v_{i+1,j+1,k} + v_{i+1,j-1,k}\right) - v_{i,j,k} \qquad (3.12)$$

3.3.2 Farneback's Estimation

Farneback[135] proposed a method that can estimate the optical flow between two frames and has been added to the OpenCV computer vision function library since version 2.4.0 for dense optical flow calculation[136]. The dense optical flow in the thesis is extracted based on Farneback's algorithm. This method estimates the neighbourhood of the frame by quadratic polynomials. The displacement between two frames can be estimated by obtaining the polynomial expansion coefficients, as described in the following section.

3.3.2.1 Polynomial Expansion

The neighbourhood of each pixel is estimated by the polynomial expansion[135]:

$$f(\boldsymbol{x}) \approx \boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^{\boldsymbol{T}}\boldsymbol{x} + c \qquad (3.13)$$

where $\boldsymbol{A}$ is a symmetric matrix, $\boldsymbol{b}$ a vector and c a scalar. The coefficients are estimated in terms of normalized convolution with the basis functions:

$$\{1, x, y, x^2, y^2, xy\} \qquad (3.14)$$

for the 2D case, which is exactly what is used here, but also generalises to higher dimensionalities. The relationship between the coefficients $\{r_i\}$, which are obtained from normalized convolution, and polynomial expansion is straightforward:

$$\begin{pmatrix} x & y \end{pmatrix}\boldsymbol{A}\begin{pmatrix} x \\ y \end{pmatrix} + \boldsymbol{b}^{\boldsymbol{T}}\begin{pmatrix} x \\ y \end{pmatrix} + c = r_1 + r_2 x + r_3 y + r_4 x^2 + r_5 y^2 + r_6 xy \qquad (3.15)$$

Then we have:

$$c = r_1, \boldsymbol{b} = \begin{pmatrix} r_2 \\ r_3 \end{pmatrix}, \text{and } \boldsymbol{A} = \begin{pmatrix} r_4 & r_6/2 \\ r_6/2 & r_5 \end{pmatrix}, \qquad (3.16)$$

3.3.2.2 Displacement Estimation

The ideal displacement can be estimated by polynomials of two frames. Consider the quadratic polynomial of one frame:

$$f_1(\boldsymbol{x}) = \boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{A}_1\boldsymbol{x} + \boldsymbol{b}_1^{\boldsymbol{T}}\boldsymbol{x} + c_1 \qquad (3.17)$$

And the new frame f_2 by a displacement of $\boldsymbol{d}$,

$$f_2(\boldsymbol{x}) = f_1(\boldsymbol{x}-\boldsymbol{d}) = (\boldsymbol{x}-\boldsymbol{d})^T\boldsymbol{A}_1(\boldsymbol{x}-\boldsymbol{d}) + \boldsymbol{b}_1^T(\boldsymbol{x}-\boldsymbol{d}) + c_1$$

$$= \boldsymbol{x}^T\boldsymbol{A}_1\boldsymbol{x} + (\boldsymbol{b}_1 - 2\boldsymbol{A}_1\boldsymbol{d})^T\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{A}_1\boldsymbol{d} - \boldsymbol{b}_1^T\boldsymbol{d} + c_1$$

$$= \boldsymbol{x}^T\boldsymbol{A}_2\boldsymbol{x} + \boldsymbol{b}_2^T\boldsymbol{x} + c_2 \qquad (3.18)$$

Let the corresponding coefficients be equal:

$$\boldsymbol{A}_2 = \boldsymbol{A}_1 \qquad (3.19)$$

$$\boldsymbol{b}_2 = \boldsymbol{b}_1 - 2\boldsymbol{A}_1\boldsymbol{d} \qquad (3.20)$$

$$c_2 = \boldsymbol{d}^T\boldsymbol{A}_1\boldsymbol{d} - \boldsymbol{b}_1^T\boldsymbol{d} + c_1 \qquad (3.21)$$

Then we can have $\boldsymbol{d}$, if $\boldsymbol{A_1}$ is non-singular:

$$2\boldsymbol{A}_1\boldsymbol{d} = -(\boldsymbol{b}_2 - \boldsymbol{b}_1) \qquad (3.22)$$

$$\boldsymbol{d} = -\frac{1}{2}\boldsymbol{A}_1^{-1}(\boldsymbol{b}_2 - \boldsymbol{b}_1) \qquad (3.23)$$

Equation 3.19 shows that every frame has the same polynomial coefficients, which is not realistic. Under this assumption equation 3.23 is still valid for real problems, although errors are introduced when the assumption is relaxed. The problem to be addressed is how to limit these errors so that the algorithm performs adequately well. Consequently, the polynomial coefficients of two frames are presented separately as $\boldsymbol{A_1}(\boldsymbol{x})$, $\boldsymbol{b_1}(\boldsymbol{x})$, c_1 and $\boldsymbol{A_2}(\boldsymbol{x})$, $\boldsymbol{b_2}(\boldsymbol{x})$, c_2. Ideally, as stated above, $\boldsymbol{A_1}$ should equal $\boldsymbol{A_2}$ according to equation 3.19, but in practice the approximation is:

$$\boldsymbol{A}(\boldsymbol{x}) = \frac{\boldsymbol{A_1}(\boldsymbol{x})+\boldsymbol{A_2}(\boldsymbol{x})}{2} \qquad (3.24)$$

and

$$\Delta\boldsymbol{b}(\boldsymbol{x}) = -\frac{1}{2}[\boldsymbol{b_2}(\boldsymbol{x}) - \boldsymbol{b_1}(\boldsymbol{x})] \qquad (3.25)$$

Then the primary constraint is:

$$\boldsymbol{A}(\boldsymbol{x})\boldsymbol{d}(\boldsymbol{x}) = \Delta\boldsymbol{b}(\boldsymbol{x}) \qquad (3.26)$$

Where *d(x)* represents the displacement in equation 3.22 is replaced by a spatially varying displacement field.

3.3.2.3 Estimation over Neighbourhood

By applying the equation above, the results achieved too noisy. The author, Farneback, added the assumption that the displacement field is slowly varying, which is very similar to Horn Schunk's smoothness constraint. Then the integration of errors between neighbours *I* of *x* is minimized:

$$min \sum_{\Delta \boldsymbol{x} \in \boldsymbol{I}} \omega(\Delta \boldsymbol{x}) |\boldsymbol{A}(\boldsymbol{x} + \Delta \boldsymbol{x}) \boldsymbol{d}(\boldsymbol{x}) - \Delta \boldsymbol{b}(\boldsymbol{x} + \Delta \boldsymbol{x})|^2 \qquad (3.27)$$

Then the minimum is obtained for:

$$\mathrm{d(x)} = (\sum \omega A^T A)^{-1} \sum \omega A^T \Delta b \qquad (3.28)$$

In practise, he computes $\boldsymbol{A^T A}$ and $\boldsymbol{A^T \Delta b}$ pointwise and average these with ω before calculating the displacement.

3.4 Expression Video Data

Each expression video clip shows the process of changing appearance when an expression is made. The optical flow over this period is extracted as a feature for expression recognition. Video clips, at 25 FPS, are from 3 seconds to 6 seconds long and each video clip contains one person sitting in front of a simple background under good illumination. In the picture the face is clearly shown in a frontal direction without interruption. The clip usually consists of a transition from a neutral expression to one of the six universal expressions (happiness, sadness, anger, fear, disgust and surprise). Some clips also capture the transition from one expression back to a neural expression, but all videos include the complete transition to one expression, and this expression is what defines the class label of the clip. Every video clips is cropped so that only the face is shown and is resized to 120 by 120 pixels. Every expression video clip is then labelled with the time tag of the starting frame of the expression and the frame at which expression is fully expressed. The optical flow is calculated within these periods as the rising parts of the expression. Each frame within this time period is compared with a frame, N frames previously. The optical flow is extracted between these frames in preparation for future classification, as shown in figure 3.3.

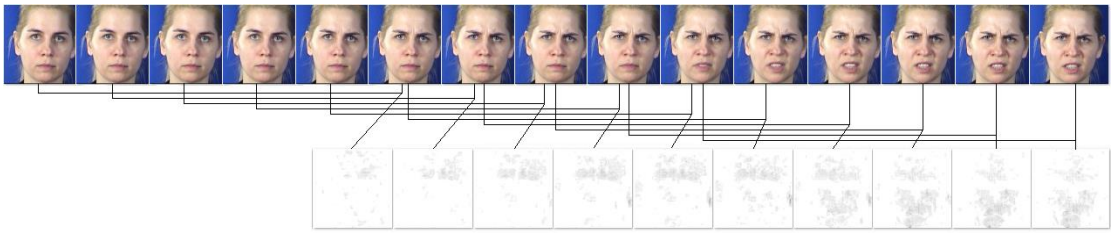


Figure 3.3: Example calculation of optical flow from a video clip

The expression commences in the sixth frame and is fully expressed by the fifteenth frame. This frame is compared with the first frame and the optical flow is calculated between the two. Other frames between the sixth and fifteenth frame are also compared with the preceding fifth frame (for N = 5). Hence, from the sixth to the fifteenth frame, there are ten sets of optical flow to consider. The minimal value of N is 2, but the velocity of flow would be too small making the feature less distinguishing in this case. When N equals 5, the flows are more obvious due to more differences between two frames. The frames before the expression are no less than 6 frames in video clips from both databases. So the N is selected to be 5.

In this way, the optical flow from the rising part of an expression is extracted. One such example is given in the following figures 3.4-3.6.

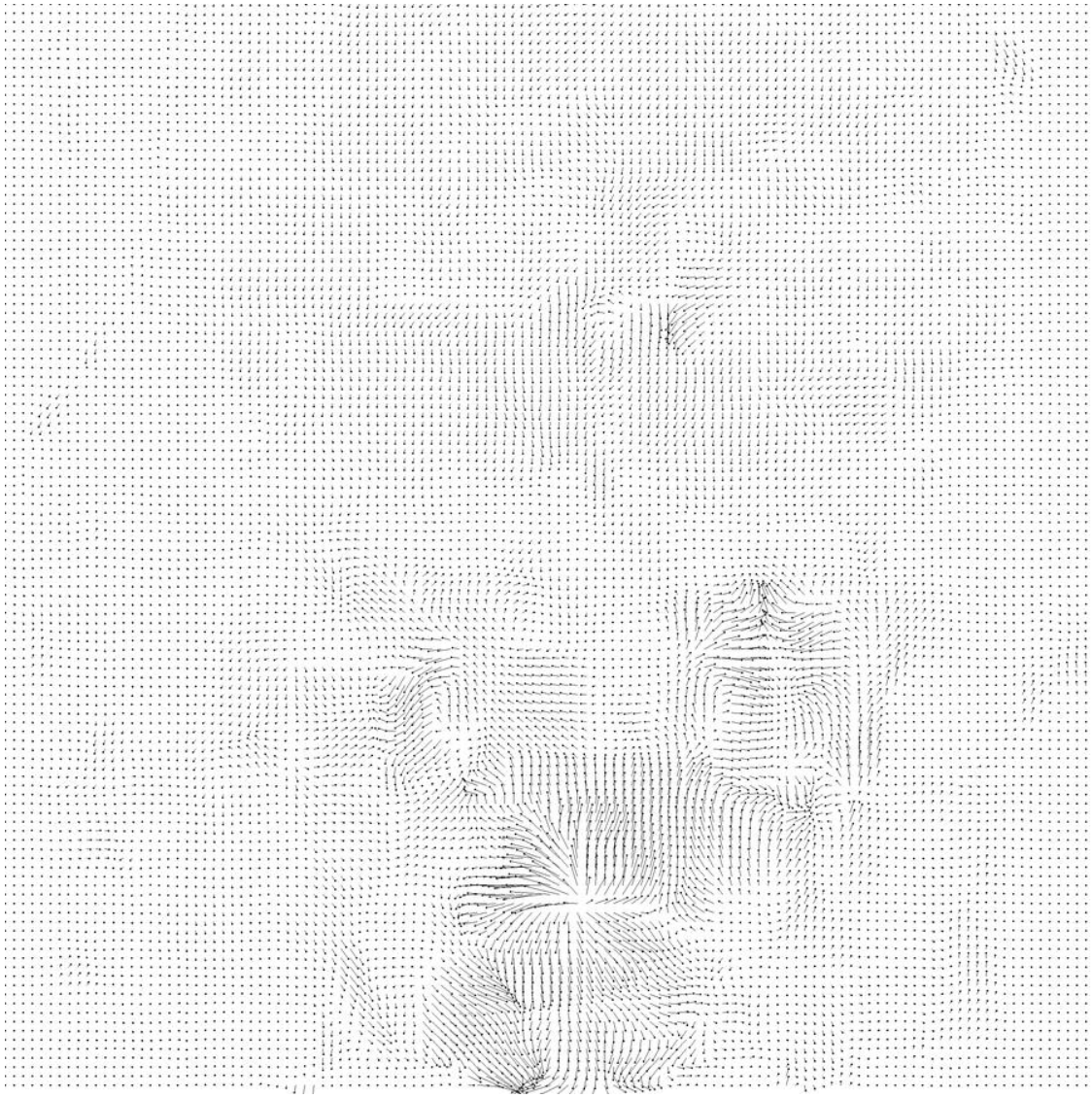


Figure 3.4: Optical Flow extracted using the Farnback algorithm

Figure 3.5: Original two frames used to calculate optical flow

Figure 3.4 shows the original dense optical flow extracted from the pixel displacements between frame 10 and frame 15, as shown in Figure 3.5. The class label of the video is the expression *angry*.

In this example of dense optical flow, considerable noise caused by illumination variance and head movement can be observed. One example is the small optical flows that occur throughout the image, even where no obvious displacement has taken place. To supress this kind of flow, a threshold has been applied, reducing the effect of noise. The threshold value used here is the magnitude of flow equal to 0.9 derived from sqrt(x^2+y^2). The flows with the magnitude less than 0.9 are made to 0, those equal or above 0.9 retain their original value:

$$T(x,y) = \begin{cases} [0,0], if\ Mag(x,y) < 0.9 \\ [x,y], if\ Mag(x,y) \geq 0.9 \end{cases} \qquad (3.29)$$

This process improves the performance of the clustering algorithm described in the following section. Figure 3.6 shows how dense optical flow performs with and without thresholding applied. The value 0.9 for the threshold is selected from 0.7 to 1.0 with 0.1 steps and considered performing better on remove the unwanted noise flows while keeping useful flows.

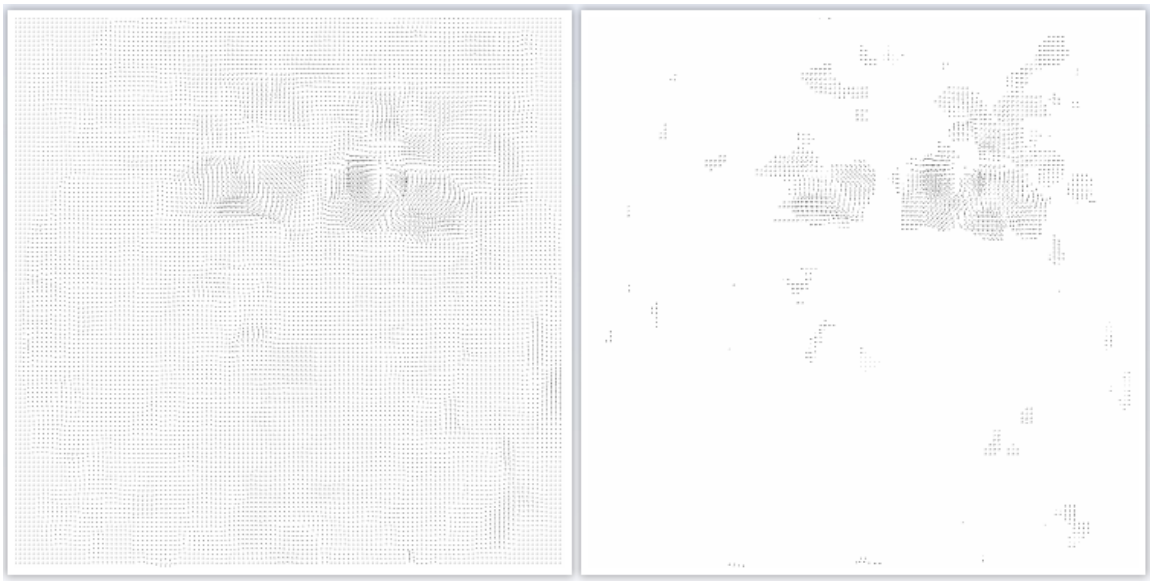


Figure 3.6: Effect of thresholding on dense optical flow

The following samples are taken from the MMI and FG-NET databases. The numbers of selected samples are shown in the tables 3.1 and 3.2 below:

Table 3.1: Details of samples taken from MMI database

Expression	Happiness	Anger	Surprise	Fear	Sadness	Disgust
Samples	304	213	224	162	165	271

Table 3.2: Details of samples taken from FG-NET database

Expression	Happiness	Anger	Surprise	Fear	Sadness	Disgust
Samples	365	371	303	113	29	292

3.5 Image Size Reduction

As mentioned above, the image has been resized to a resolution of 120 by 120 pixels, totalling 14400 pixels. Each pixel has a 2-dimension flow velocity of magnitude and direction, a two element vector in the x and y axis, represented by a two channel matrix with size of 120 by 120.

The resulting vector contains 28800 elements, which is too long for a classifier to process effectively and therefore some form of pre-processing is required to reduce the vector's size. Three methods for reduction are given in following sections.

3.5.1 Clustering as a Means of Reducing Vector Size

Facial movements associated with expression usually happen within a small area and the flows have similar velocities. If the velocity and location of each centre of the area of movement can be extracted, it may provide a good representation of the whole flow of the image. Centroid-based clustering algorithms were investigated to achieve such a representation. K-means Clustering algorithm[137] was applied for this purpose.

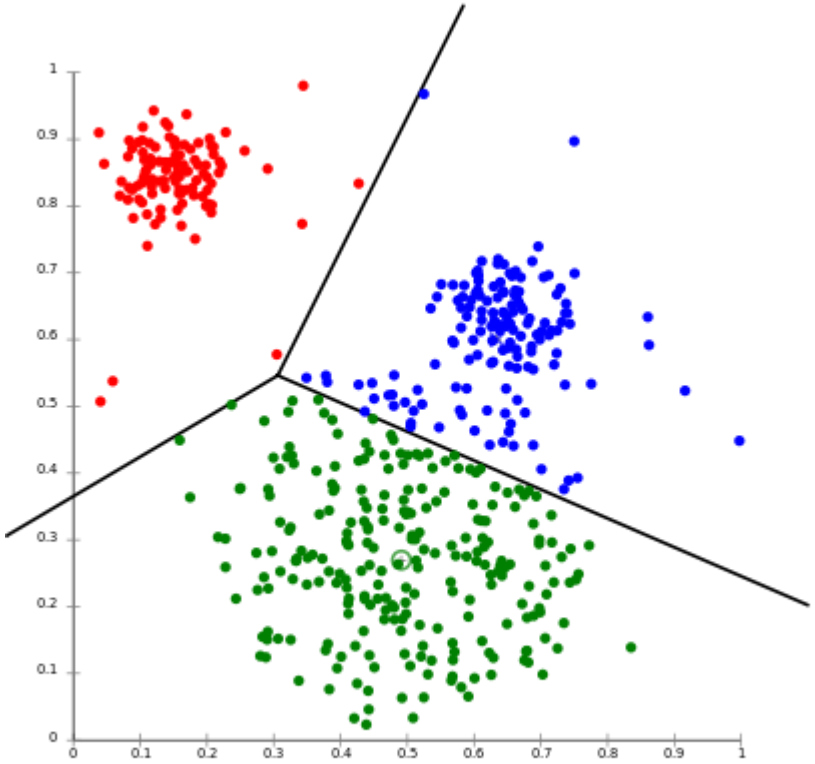


Figure 3.7: K-means Clustering Algorithm[138]

However, some problems were experienced during implementation of the clustering algorithm. But centroid-based clustering algorithm initialises with random centres for each cluster, and therefore, it returns different results each time implemented, even when using the same data. The effect of this problem will be discussed in section 3.4.1.4 - experiments and results. Other discussions and comparisons are given in following paragraphs.

3.5.1.1 Comparison of 3 Features Clustering and 4 Features Clustering

Each pixel in the image is represented by its x, y position and x, y flow velocities of direction and magnitude. An attempt is made to cluster pixels based on all 4 values (as shown in Figure 3.8) and then use the centres of these clusters to form the reduced feature vector. An attempt was also made using only 3 values (as shown in Figure 3.9): x, y and magnitude of flow, ignoring the direction of flow. In this way, the vector length reduces to 100 (25 clusters multiply 4) or 75 (25 clusters multiply 3).

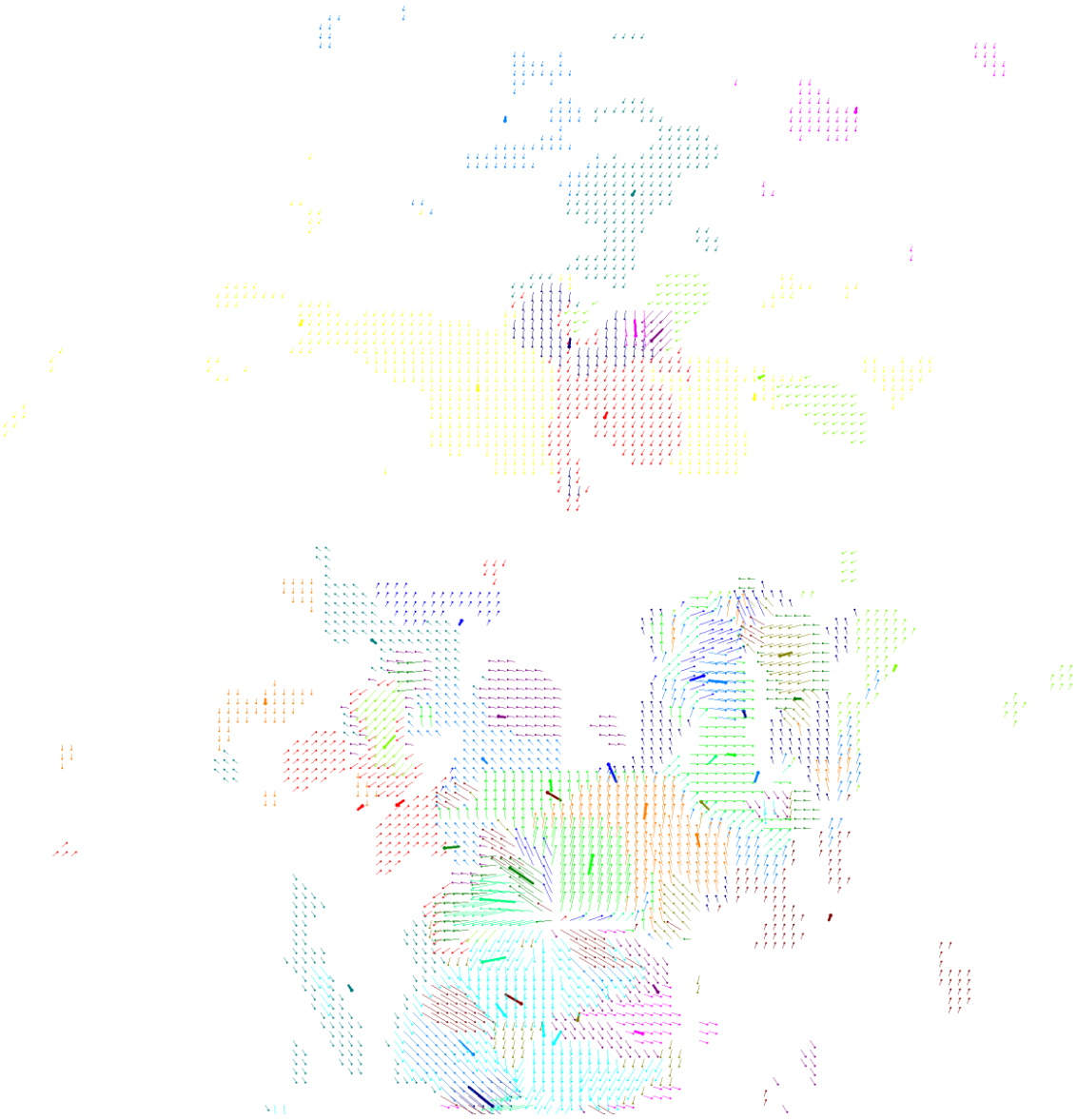


Figure 3.8: Clustering algorithm using 4 values (x, y, velocity x and velocity y). The arrows with same colour and velocity belong to one cluster. The thick arrows represent the centres of clusters.

Figure 3.9: Clustering algorithm using 3 values (x, y and magnitude of velocity). The number is the magnitude at the location where it shows. The numbers with same colour and value belong to one cluster. A bold number represents the centre of the cluster.

In figure 3.8 and figure 3.9, the results for two ways of clustering data are presented. The method using 3 values has fewer vector elements than that with 4 values. If the cluster number is 25, then the 3-value method has in total 75 numbers in the vector. Of the 25 cluster centres, each has 3 values: x, y coordinates and magnitude. Similarly, the 4-value method has 100 numbers representing 25 clusters. Each cluster centre has 4 values: x, y coordinates and x, y velocities. Clearly, the 3-value method one is 25% smaller than the 4-value method. This advantage is considerable, but also has drawbacks. The 3-value method lacks information of direction. If the mouth corner moves upwards or downwards, only the magnitude is extracted and the direction is unknown. In this case, it is not possible to distinguish the flows between each case. Therefore, it may be that directional information cannot be ignored for the sake of saving vector space and, therefore, 4 value feature method is retained.

3.5.1.2 Weights for Position Coordinates and Flow Velocity

This section is going to discuss the effect of employing different weights for position coordinates and flow velocity.

Consequently, the 4-value vector method was selected for the clustering algorithm. x and y coordinates are in same coordinate references, so the weight between x and y is simply 1:1. It is the same situation regarding the x and y velocities, but a weighting between location and velocity is required as it affects the clustering results considerably. If the velocity component is assigned a greater weight, flows with similar velocities are considered as one cluster, even if they are spatially set apart in the image; if values of location are assigned a greater weight, flows within an area will count as one cluster regardless of what velocities (magnitude and direction) they may have. Figures 3.11 to 3.13 show the results of clustering with different weightings applied to the optical flow shown in Figure 3.10.

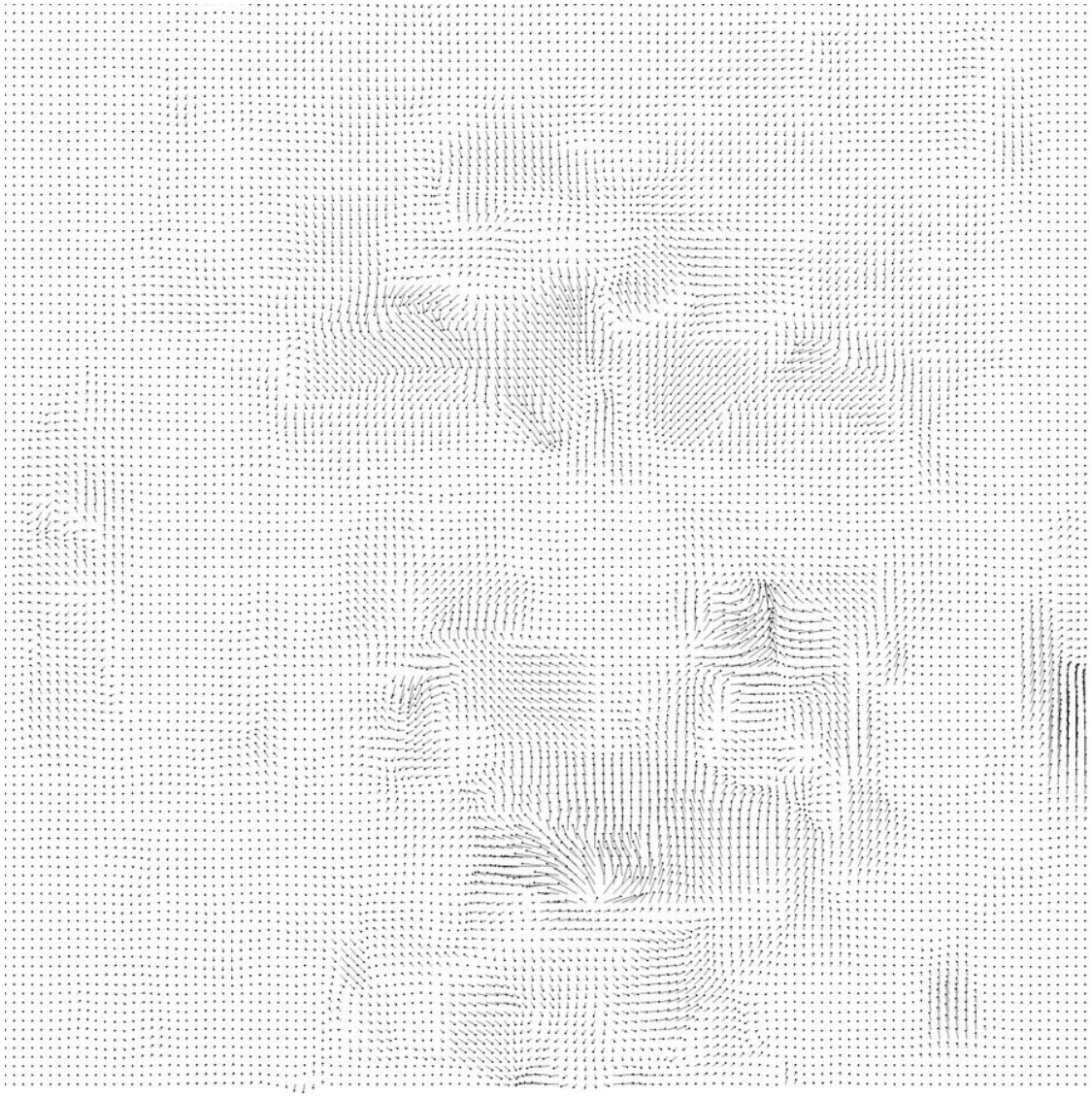


Figure 3.10: Optical Flow extracted using the Farnback algorithm

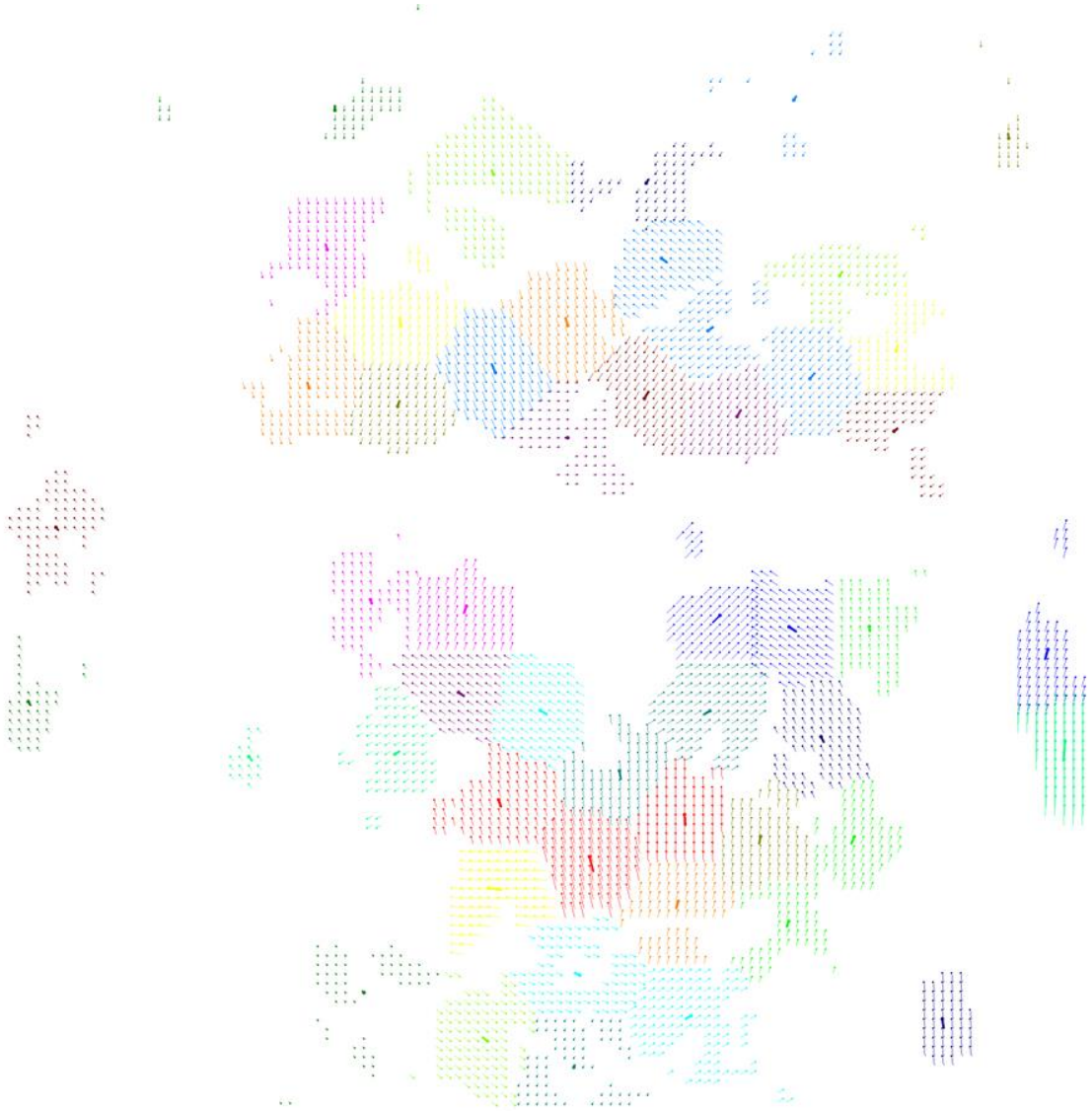


Figure 3.11: Clustering resulting from equal weighting (1:1) of location and velocities (range of x and y coordinates: 0 - 120)

In Figure 3.11, the location values and velocities have the same weight. The results pictured show that the clusters are generally arranged in blocks, indicating that the vector means have similar location values and are considered as one cluster, but the velocities have been largely ignored. The block size depends on the number of clusters - the more clusters the smaller the block size. The result of this weighting has placed too much emphasis on positional information, so when all elements in one cluster are replaced by the cluster centre value, the result is quite different from the original optical flow (as seen in figure 3.10) and arguablely not representative in a meaningful way.

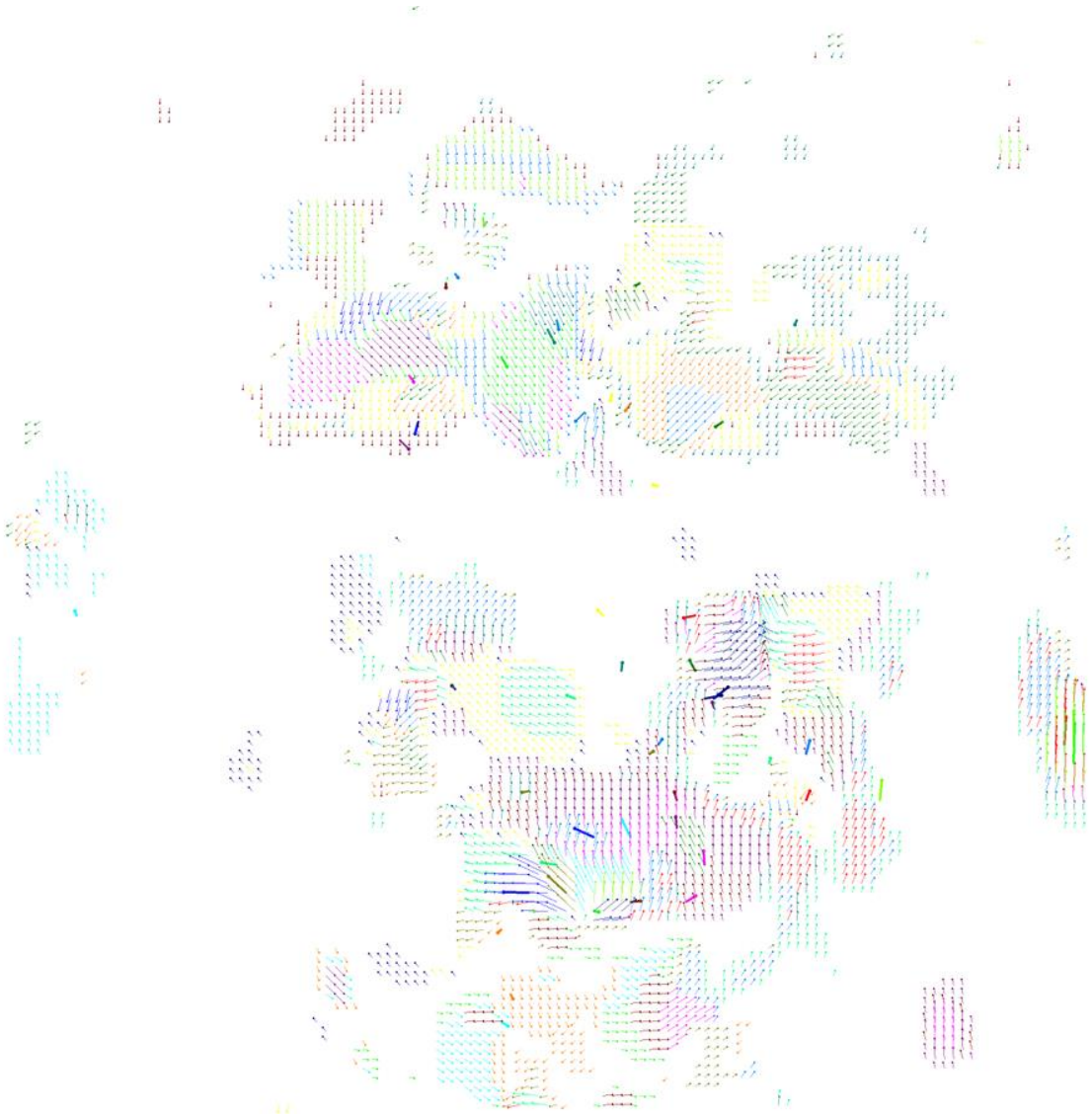


Figure 3.12: Clustering resulting from a weighting of 1:120 for location and velocities (range of x and y coordinates: 0 - 1)

In figure 3.12, the weighting of velocity is much higher than that for location information. The clustering algorithm therefore considers the flow velocity to be more important than its location. Consequently, flows that have similar velocity are grouped in the same cluster even if they are spatially apart. This means the cluster centre may be allocated in a position away from the contributing clusters, which arguably means the centres do not represent the flow distribution effectively.

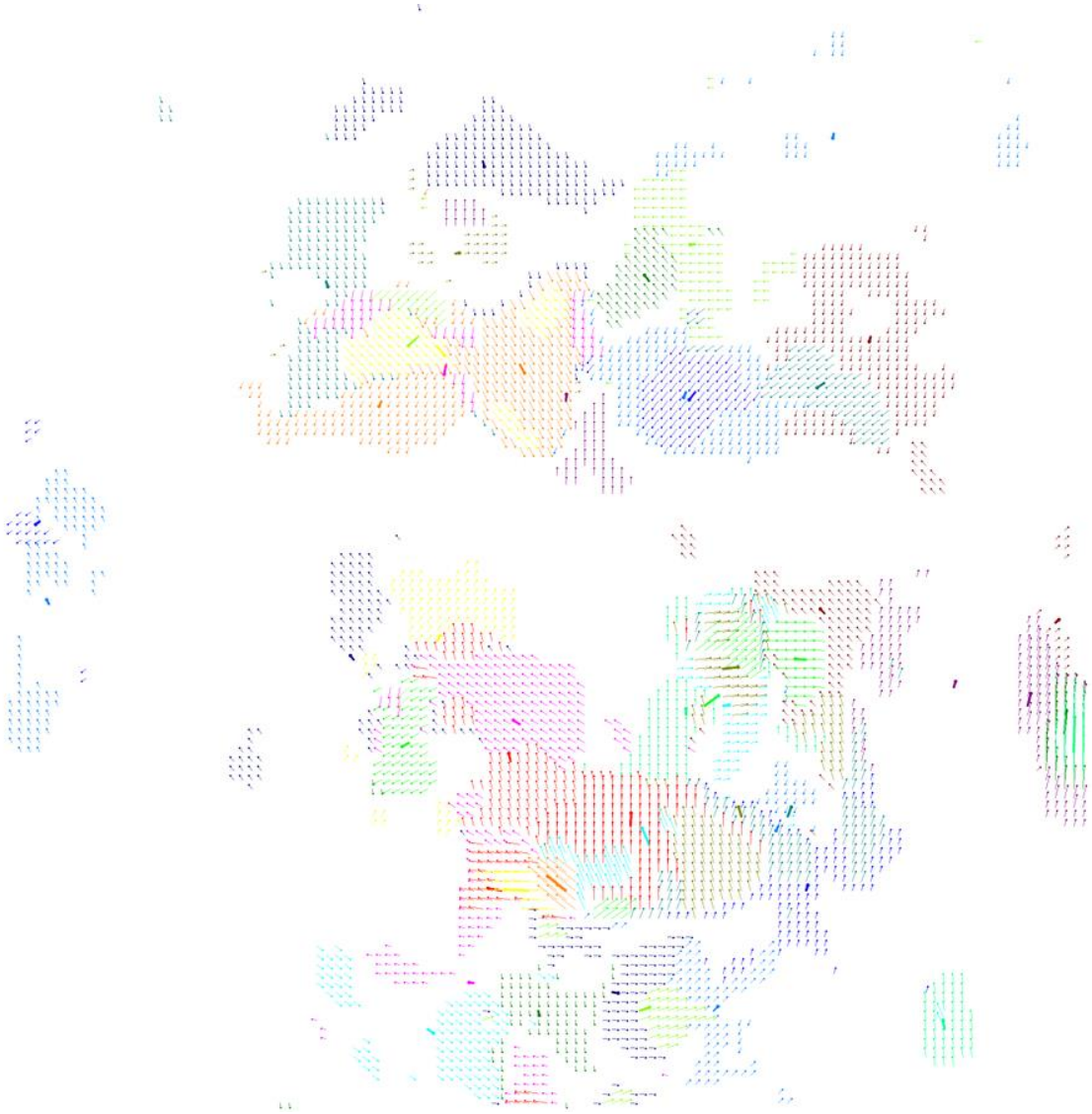


Figure 3.13: Clustering resulting from a weighting of 1:24 for location and velocities (range of x and y coordinates: 0 - 5)

This weighting of 1:24 for velocity and location (as shown in Figure 3.13) is used for all following experiments as it is considered to provide an optimal balance between location and velocities. The weight is derived from the proportion of magnitude values of optical flow and the coordinate values. From figure 3.9 the maximum magnitude for a cluster centre is 7.77 and other large flows range from 4 to 6. The weighting is taken from the number of pixels in one dimension (120) to the average major magnitude of optical flow (approximately 5). Clusters derived with this weighting are therefore dependent on both position and velocity in equal measure and the position of the cluster centre still sits within corresponding cluster.

3.5.1.3 Number of Clusters

Flows in one cluster may be widely dispersed and yet the centre of cluster will be located at the average position of these flows. To avoid this problem, the number of clusters can be increased so widely dispersed flows will form separate clusters. There is still a trade off because additional clusters enlarge the vector length as a result. Figure 3.14 shows the original optical flow and figures 3.15 and 3.16 show the results of specifying 100 clusters and 20 clusters, respectively, from which the difference can be seen clearly.

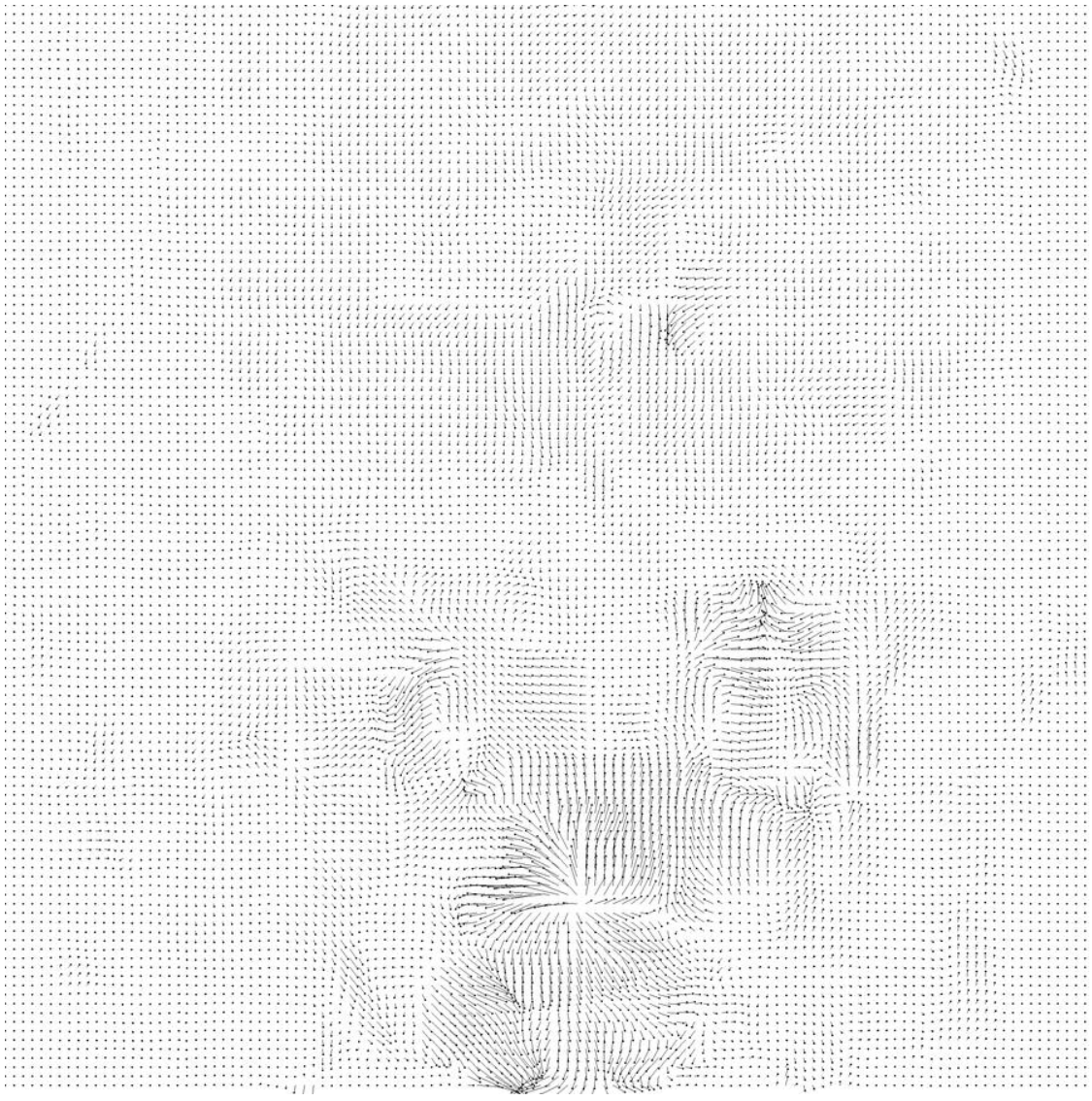


Figure 3.14: Optical flow extracted by Farnback algorithm

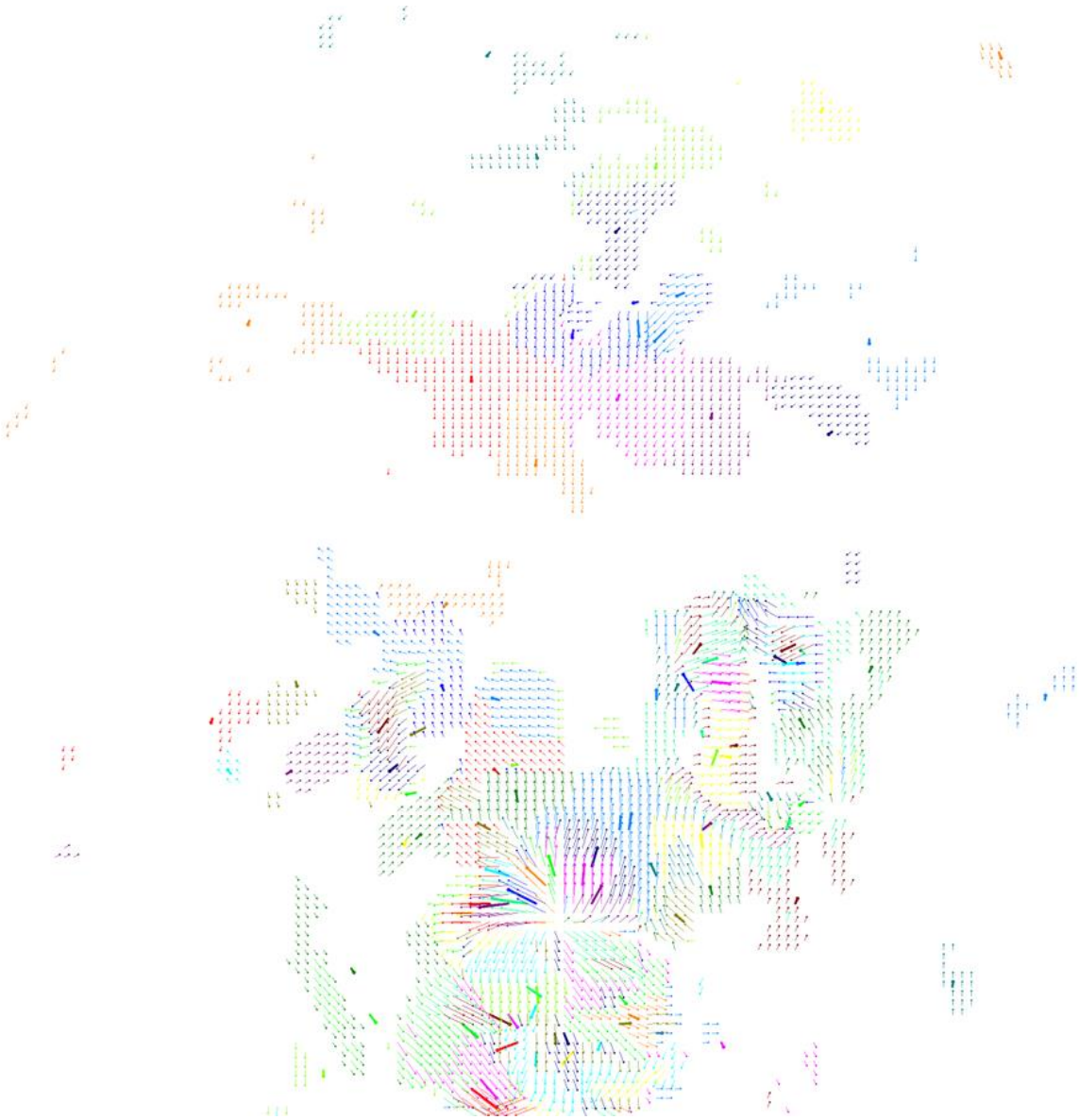


Figure 3.15: Clustering the optical flow using 100 clusters

In the figure 3.15, the weighting between coordinate and flow velocity is chosen to be 1:24, which is the better weighting method described from the previous section - all following clustering uses this weighting. The results show the flows after clustering are very accurate and the centres of clusters are almost always within the corresponding clusters. The larger the cluster number, the better the representation of the original optical flow pattern. However, the drawback is also clear - the vector length is 400 (100 * 4), which is still too long for effective classification by the methods considered in this thesis. Especially for the EAs, the increasing dimensions of searching space results an exponential growth of the procedure time, and also the difficulty to find the optimal solution.

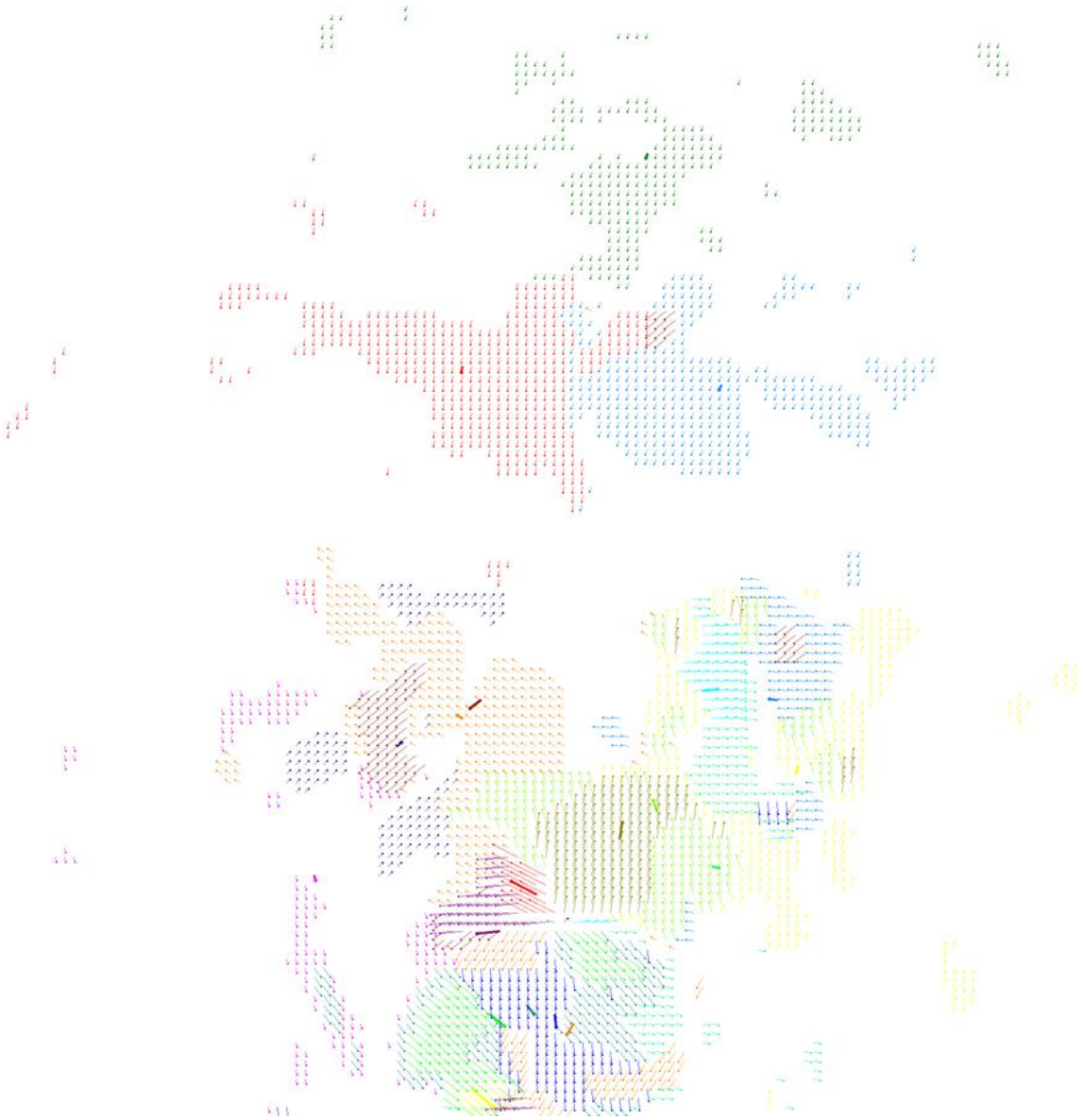


Figure 3.16: Clustering the optical flow using 20 clusters

In Figure 3.16, the number of clusters is small and so the area of each cluster is comparatively large. The area is not always continuous and flows in one cluster can have a range of different velocities, or have similar velocity but are located far apart.

The trade-off between vector length and clustering performance is a compromise between achieving a clustering result that adequately represents the original flow and a vector length that is acceptably small. To address this, the number of clusters was set to 50, which is chosen from all the tens between 20 and 100 and balanced clustering result and vector length. The length is further reduced by choosing 25 cluster velocities randomly for the subsequent classification stage. Although this achieves representative cluster results and desired vector length, accurate classification may still be compromised by the loss of information incurred in the process.

3.5.1.4 Experimentation and Results

The following experiments are based on the K-means clustering algorithm[138] with a threshold set to 0.9, number of clusters to 50 from which 25 are selected for classification, a weighting between coordinates and flow velocity of 1:24, resulting in a vector length of 100. The classifier used is Support Vector Machine (SVM) (C_SVC, RBF)[70, 71] with the default setting of OpenCV[139]. The experiments are based on MMI database. Because of the random initial cluster centre, the clustering algorithm returns a different result each time it is run. This results in different cluster distributions, location and ordering of cluster centres, as well as the cluster centres selected for the classification stage. Therefore, to provide a more representative sampling of the original flow data, the clustering procedure is repeated a number of times.

In order to investigate the classification of the six universal expressions, two sets of experiments were undertaken, all of which were binary classification tasks. The first set of experiments is designed to distinguish between any two expressions from the six on a pair-wise basis, making 15 classification tasks in total. The second set of experiments compares each expression against all other expressions as a single set, making six classification tasks in total.

When considering the number of samples for each expression, it is apparent that there are considerably fewer for sadness and fear, than for the other expressions. Specifically, the data for training the classifiers is imbalanced, by a factor of approximately 1:5, which will affect classifier performance. To address this imbalance, for some experiments, repeat samples were added to these two expression classes to bring the number of samples to that of other classes. Where this has been used, it is referred to as the *balanced* or *oversampled* dataset.

In summary, two sets of experiments were performed – (i) pair-wise comparison of each expression and, (ii) the individual comparison of each expression against a set comprising all other expressions. For each, both original (imbalanced) and balanced datasets were used in different combinations for training and testing SVM classifiers. In all cases, the MMI database was used.

Experiment (i): Pair-wise comparison of each expression

Table 3.3 shows the results of training the SVM classifier for pair-wise comparison of expressions using the original data set, and testing it using the balanced dataset. Table 3.4 shows the results of the SVM classifier trained original dataset and tested with re-clustered data, and Table 3.5 results of the SVM classifier trained balanced dataset and tested with re-clustered data. This approach demonstrates good performance for some expression pairs, but is not effective for others. Although the training data and test data are different, because of the randomness of clustering algorithm, they originate from the same video clips and therefore is not a good test of how the classifiers will perform on unseen clips.

Table 3.3: Pair-wise comparison of expressions trained with original dataset (imbalance) and tested with balance dataset

Train with small set, test with big set (over sampled)						
Positive \ Negative		Anger	Surprise	Fear	Sadness	Disgust
Happiness	TPR % TNR %	93.55 82.54	82.89 72.23	84.67 69.51	88.29 77.7	86.71 67.53
Anger		TPR % TNR %	88.64 75.98	83.94 78.02	76.62 60	74.37 72.92
Surprise			TPR % TNR %	80.45 53.95	79.29 77.09	68.93 76.83
Fear				TPR % TNR %	63.95 78.91	53.47 81.92
Sadness					TPR % TNR %	45.33 83.32

Table 3.4: Pair-wise comparison of expressions trained with original (imbalance) dataset

Test with re-clustered data, small set						
Positive \ Negative		Anger	Surprise	Fear	Sadness	Disgust
Happiness	TPR % TNR %	94.74 84.98	98.68 91.96	97.7 93.21	100 100	91.45 72.69
Anger		TPR % TNR %	89.67 76.79	96.71 92.59	94.37 82.42	96.24 91.51
Surprise			TPR % TNR %	95.09 96.91	87.05 82.42	91.07 99.26
Fear				TPR % TNR %	91.36 97.58	89.51 99.63
Sadness					TPR % TNR %	83.03 97.42

Table 3.5: Pair-wise comparison of expressions trained balance dataset

Test with re-clustered data, big set(over sampled)						
Positive \ Negative		Anger	Surprise	Fear	Sadness	Disgust
Happiness	TPR % TNR %	93.16 86.85	98.03 90	97.96 87.16	92.57 75.39	96.64 90.11
Anger		TPR % TNR %	100 99.91	100 99.88	93.62 80.12	95.68 87.45
Surprise			TPR % TNR %	85.63 57.53	100 100	99.46 100
Fear				TPR % TNR %	99.88 100	81.36 99.19
Sadness					TPR % TNR %	80.85 98.3

Results show that the performances of SVM are greatly affected by the ratio of training data. The results of imbalanced training data are much worse than those of balanced training data.

Experiment (ii): Comparison of each individual expression against a set comprising all other expressions

Table 3.6 shows results for each expression compared against the set of all other expressions with the original (imbalance) dataset. The first column reports the training fitness in terms of both true positives (*positive* or *TP)* and true negatives (*negative* or *TN*). The second column reports the performance of the trained classifier when tested with re-clustered data, again in terms of TP, TN and the respective true positive and true negative rates (*TPR*, *TNR*). Finally, the third column reports results for the trained classifier tested with the balanced (oversampled) dataset. The results are poor due largely to the imbalance of the training data.

Table 3.6: Individual comparison of expressions trained with original (imbalance) dataset

Train with imbalanced data												
	Training data		Test with re-clustered data				Test with oversampled data					
Happiness	P	304	TP	251	TPR	82.57%	P	7600	TP	3777	TPR	49.70%
	N	1035	TN	1021	TNR	98.65%	N	5175	TN	4856	TNR	93.84%
Anger	P	213	TP	146	TPR	68.54%	P	5325	TP	1643	TPR	30.85%
	N	1126	TN	1116	TNR	99.11%	N	5630	TN	5433	TNR	96.50%
Surprise	P	224	TP	17	TPR	7.59%	P	5600	TP	117	TPR	2.09%
	N	1115	TN	1115	TNR	100.00%	N	5575	TN	5569	TNR	99.89%
Fear	P	162	TP	74	TPR	45.68%	P	4050	TP	22	TPR	0.54%
	N	1177	TN	1177	TNR	100.00%	N	5885	TN	5881	TNR	99.93%
Sadness	P	165	TP	0	TPR	0.00%	P	4125	TP	0	TPR	0.00%
	N	1174	TN	1174	TNR	100.00%	N	5870	TN	5870	TNR	100.00%
Disgust	P	271	TP	110	TPR	40.59%	P	6775	TP	207	TPR	3.06%
	N	1068	TN	1068	TNR	100.00%	N	5340	TN	5335	TNR	99.91%

Table 3.7 shows results for a similar experiment, but with a small balance training set. The results are much improved over those using the original (imbalance) dataset, but results are still poor for some expressions.

Table 3.7: Individual comparison of expressions trained with balanced training dataset

Train with balanced data (over sampled)												
	Training data		Test with re-clustered data				Test with oversampled data					
Happiness	P	1520	TP	1414	TPR	93.03%	P	7600	TP	6354	TPR	83.61%
	N	1035	TN	772	TNR	74.59%	N	5175	TN	3696	TNR	71.42%
Anger	P	1065	TP	899	TPR	84.41%	P	5325	TP	4209	TPR	79.04%
	N	1126	TN	891	TNR	79.13%	N	5630	TN	4312	TNR	76.59%
Surprise	P	1120	TP	1114	TPR	99.46%	P	5600	TP	3963	TPR	70.77%
	N	1115	TN	1115	TNR	100.00%	N	5575	TN	4316	TNR	77.42%
Fear	P	810	TP	673	TPR	83.09%	P	4050	TP	1947	TPR	48.07%
	N	1177	TN	1163	TNR	98.81%	N	5885	TN	4907	TNR	83.38%
Sadness	P	826	TP	701	TPR	84.87%	P	4125	TP	2145	TPR	52.00%
	N	1174	TN	1086	TNR	92.50%	N	5870	TN	4564	TNR	77.75%
Disgust	P	1355	TP	1321	TPR	97.49%	P	6775	TP	4662	TPR	68.81%
	N	1068	TN	1036	TNR	97.00%	N	5340	TN	3345	TNR	62.64%

Compare the results in table 3.6 and 3.7, the influence of imbalanced training data on SVM can be observed easily. The results in table 3.7, which are trained by balanced data, are much better than those in table 3.6, which are trained by imbalanced data. The true positive rates of surprise, fear, sadness and disgust are even lower than 5%. True positive rates increase significantly in table 3.7, but fear and sadness still have much lower rates than other expressions.

3.5.1.5 Summary

The randomness affects the data. It is effective way to use the randomness to enlarge the data set for a balanced data set. This helps to overcome the problems that imbalanced data bring in classification. Simply repeat the data without randomly differences is not as good as randomness for balancing data set. But the randomness should be used in a proper way. It would be not a problem, if the randomness happens on the slightly changing of the magnitude value at a smiling mouth corner. Because the meaning does not change, it is still the magnitude of flow velocity. The slight difference on value will not change the fact of the man is smiling. But the order of how vectors are arranged is important for classification. Consider the K-means (or any other classifiers), the input data contains two elements which stand for x and y locations. If the order of x and y are randomly changed before passing to the classifier, it would definitely not return a good classification result. As discussed in the beginning of 3.4.1.4, the order of cluster centres will change during clustering algorithm. The locations of coordinates and velocities are confirmed not going to change. But the order of cluster centres will change; this is the reason why this method cannot have good testing results even on training data. So this cluster feature used here is not effective for expression recognition.

Clustering algorithms other than K-means have also been considered like hierarchical clustering, but it does still not provide good results because the random order of the meaning of cluster centres for classification. The clustering algorithm is finally abandoned.

A notable thing is in this experiment the same training data is used for testing. This is meaningless if it achieves excellent performance. However it is a way to disprove this method does not work well and the clustering algorithm is not used indeed.

3.5.2 Five Highest Regions of Flows Selection Method

3.5.2.1 Introduction

Following the disappointing performance of the clustering algorithm described in section 3.5.1, a closer examination of the flows and the centres of clusters from those results was conducted. The areas in which high optical flow occurred are small in number and certainly less than the 50 clusters previously used. Although it was hypothesised that small optical flow may hold important and discriminating characteristics, it was decided to concentrate on the more prominent flows, from as few as five areas within the image. Hence, the method proposed in this section is based on extracting simpler features comprising higher average flows, rather than attempting to cluster a large number of flows.

To facilitate this method, the presentation of flow is converted from Cartesian to polar coordinates, which still results in four values for each pixel: x, y, magnitude and theta. Next a windowing operation is applied, to scan the whole image, storing the five sub-windows which had highest average flow magnitude. Because the flows have been thresholded in the preprocessing stage, large areas of small flows report as having zero flow. Consequently, when calculating the average magnitude of flow within a window, zero magnitude value flows are ignored so not to penalise sparse flow magnitude values that are reported.

Next, each of five sub-windows are divided into 3 by 3 grids; each grid has an average magnitude and theta. The averages are obtained in the same way as for the average high magnitude as described above and shown in figure 3.17. Features are formed from the values of magnitude and theta taken from the nine flows in each of the five windows. These are then ordered to form a feature vector from the largest average magnitude to the smallest. The vector length is 3*3*2*5 = 90, but does not include location information.

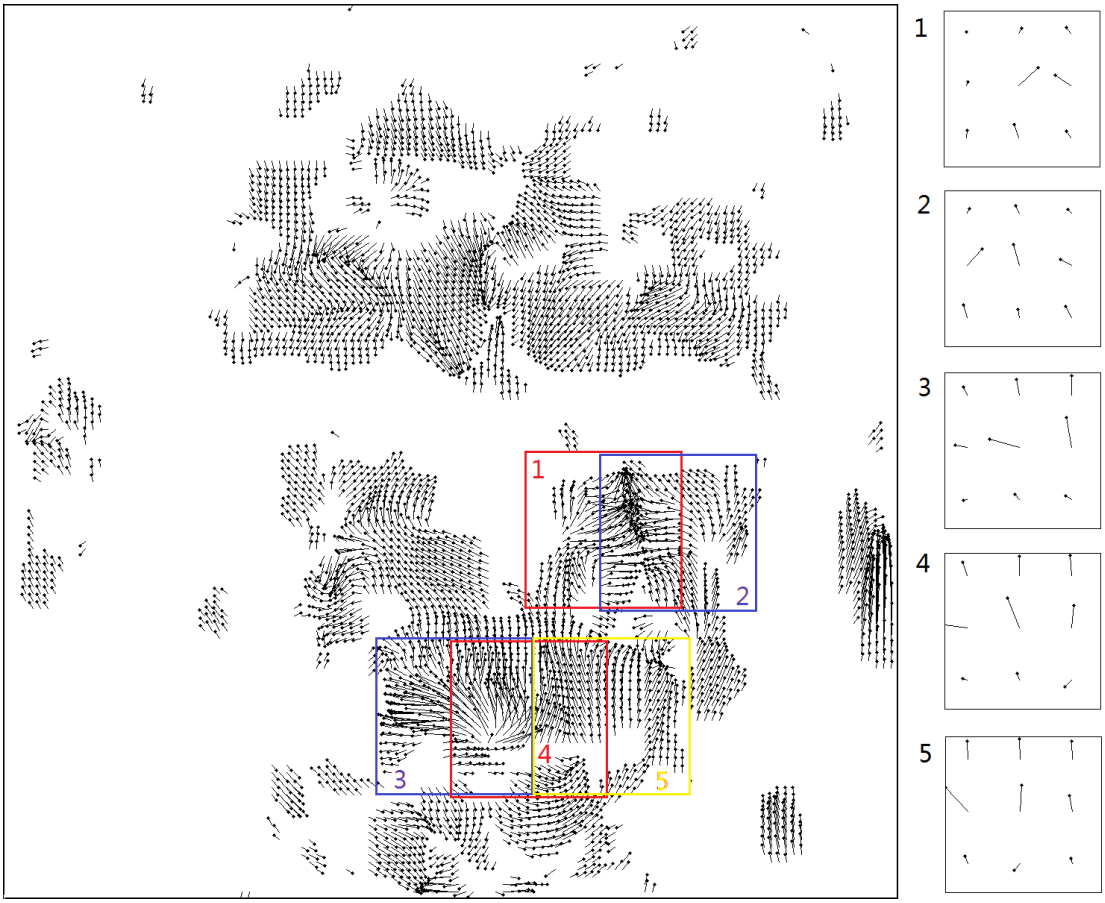


Figure 3.17: Optical flow showing the five sub-windows having highest average flow magnitude. The arrows in the sub-windows on the right are the resulting features

3.5.2.2 Experiments and Results

The following experiments use both the MMI and FG-NET databases and include six binary classifications for each expression. As this method does not include random procedures, fixed features can be extracted from the same original flow data. No oversampling or down-sampling to increase minority class data or decrease majority data is applied, so the training data for this experiment is imbalanced. However, cross validation has been applied in the training phase of the SVM to obtain the best parameters. All training data was separated to K groups and each run takes K-1 groups of data as one sub-training fold and the rest is used for testing. This K-fold cross validation is run K times for each set of SVM parameters. The total error of all K runs with the resulting set of parameter is recorded, with each possible parameter combinations will do the cross validation separately. The parameters with lowest error on K-fold cross validation, and hence, that supposed to be most suitable for the problem, is used for final training of the whole training dataset. Each binary classification experiment, happiness to non-happiness etc., the training data are all from one database (MMI or FG-Net), and the testing data are from the very

other unused database. The numbers of samples for each experiment are shown in the following table, which is generated from table 3.1 and table 3.2.

Table 3.8: Number of samples for binary classification experiments

			Happiness	Anger	Surprise	Fear	Sadness	Disgust
Samples	MMI	P	304	213	224	162	165	271
		N	1035	1126	1115	1177	1174	1068
	FG-NET	P	365	371	303	113	29	292
		N	1108	1102	1170	1360	1444	1181

The SVM classifier was trained with each database by this method and then tested with the other. The accuracies for the six expressions are shown in table 3.9. The results include the best average error rate (Err. R, equation 3.30), true positive rate (TPR, equation 3.31) and true negative rate (TNR, equation 3.32) of the 10 groups (10-fold) in cross validation for the training set. Similarly, rates are also shown for the SVM classifier test results.

$$\text{Error rate} = 1 - (TP+TN) / (P+N) \quad (3.30)$$

$$\text{True positive rate} = TP / P \quad (3.31)$$

$$\text{True negative rate} = TN / N \quad (3.32)$$

where P and N are the numbers of positive and negative samples in an experiment. TP and TN are both correct predictions, but for positive and negative respectively.

Table 3.9: SVM trained and tested through the high flow sub-window method

		Train MMI		Train FG-NET	
		Group Ave.	Test FG-NET	Group Ave.	Test MMI
Happiness	Err.R.	16.34%	20.71%	17.18%	23.53%
	TPR	47.29%	31.78%	52.46%	52.79%
	TNR	94.75%	94.95%	92.87%	83.21%
Anger	Err.R.	12.30%	24.85%	18.60%	18.45%
	TPR	35.47%	17.79%	43.50%	23.85%
	TNR	97.58%	94.46%	94.58%	92.41%
Surprise	Err.R.	12.86%	22.81%	18.33%	19.75%
	TPR	45.43%	11.55%	31.06%	14.78%
	TNR	95.64%	94.19%	94.76%	93.37%
Fear	Err.R.	11.02%	2.58%	6.92%	12.85%
	TPR	14.27%	6.90%	15.90%	0.00%
	TNR	100.00%	99.24%	99.78%	99.75%
Sadness	Err.R.	10.64%	2.58%	1.02%	12.49%
	TPR	17.60%	6.90%	37.83%	0.00%
	TNR	99.83%	99.24%	100.00%	100.00%
Disgust	Err.R.	17.49%	22.74%	16.98%	21.71%
	TPR	18.97%	2.74%	21.68%	2.52%
	TNR	98.66%	95.68%	98.19%	97.45%

The SVM parameters used for training are selected from the one with lowest error after cross validation. The parameters for all 12 SVM classifier trainings are shown in table 3.10 below.

Table 3.10: SVM parameters selected by best cross validation results

SVM type: C-SVC		Kernel: RBF					
		Happiness	Anger	Surprise	Fear	Sadness	Disgust
C (10^1)	MMI	0.25	0.25	1.25	0.25	0.25	0.25
	FG-NET	1.25	0.25	1.25	1.25	0.25	1.25
Gamma (10^-3)	MMI	2.25	33.75	2.25	33.75	33.75	33.75
	FG-NET	2.25	33.75	2.25	33.75	33.75	33.75

From the table of results, it can be seen that the imbalanced training data affects the results considerably, as the accuracies of majority classes are much higher than that for minority classes. Too few positive samples for training tends to a negative response from the classifier. Figure 3.18 below illustrates how imbalanced training data can make the boundary of the SVM shift towards the smaller data region, Where (a) is the actual segmentation of two classes, (b) shows the samples available for training, (c) the boundary set by the SVM algorithm and, (d) the classifier result and ground truth.

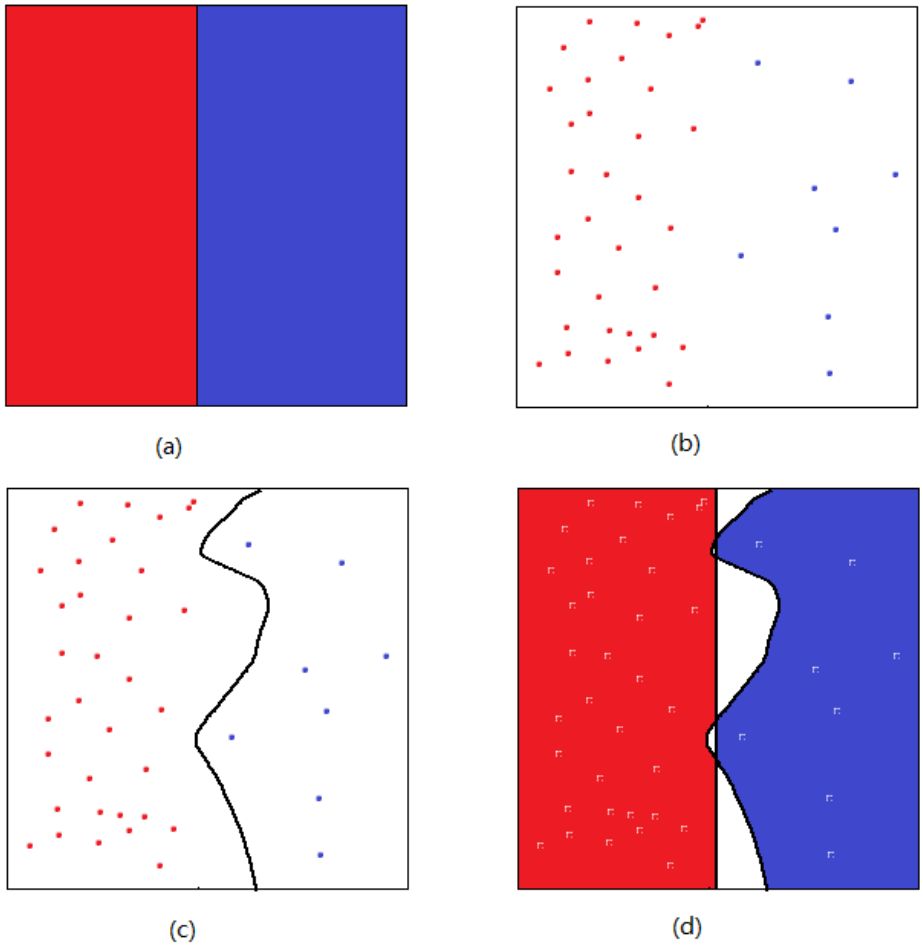


Figure 3.18: The effect of training with an imbalanced dataset

If red is negative, blue is positive, then the red area in 3.18 (d) is true negative, the blue area is true positive, the white area in the left half is false positive and the white area of right half is false negative. From the figure, false negative will be large when the positive sample for training is small. This is a simple linear example. In reality, most of the problems are non-linear, so the imbalanced classification will have more effect.

Considering the average error rate of 10 groups in table 3.11, the true positive rates are not good at under, or around 50%, which is already the smallest error of results possible among other possible parameters. It suggests high flows used in this sub-windows method are not sufficiently effective. One reason could be that the features do not include the location information of the 5 sub-windows.

If the comparison is made between the group average error column and test with the other database values, the true positive rates drop sharply, except for classification of the happiness emotion. This means the differences between the two databases of the other five expressions are significant. It is not an easy task to train and test with different databases, because the differences between databases are likely to be too large.

3.5.2.3 Summary

The results presented above suggest that lack of location information and imbalanced datasets significantly affect the accuracy and to train and test with different databases is another difficult task.

3.5.3 Average of Flows in Grid

3.5.3.1 Introduction

From the previous two feature selection methods, it was concluded that the features should contain both velocity and location information, and have the same meaning in the relative position within the resulting vector (i.e. ordered). This section introduces an algorithm to achieve these goals.

The method divides the image (of 120 x 120 pixels) into a 6 x 6 grid, each cell consisting of 20 x 20 pixels. Excluding the side columns, a 6 x 4 grid remains as shown in fig. 3.19. The average velocities of the remaining cells form the vector of length 48. (6 rows * 4 columns * 2 velocity x, velocity y = 48). The results for this feature are obtained in the same way as for the previous method.

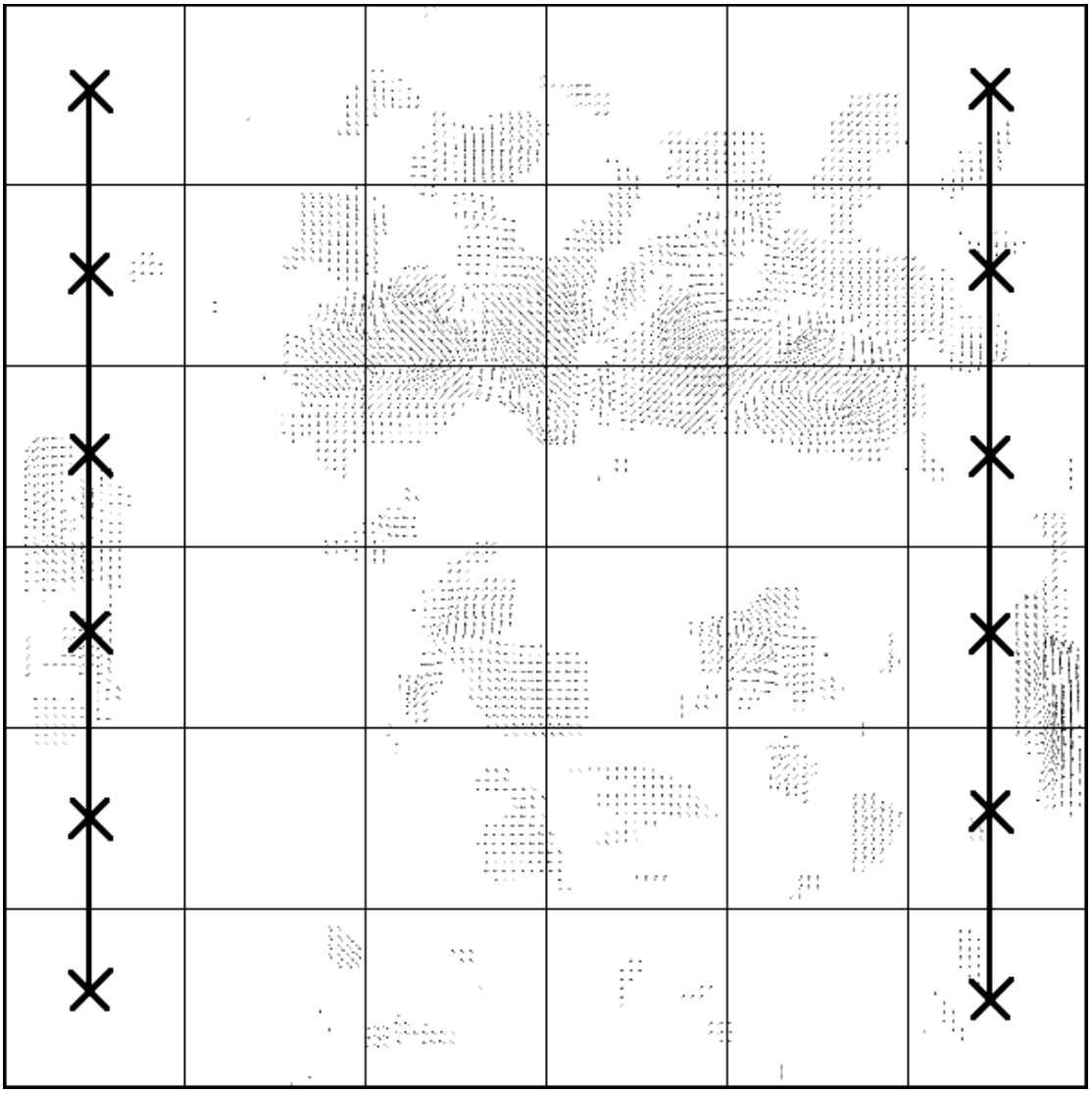


Figure 3.19: Grid used in the *average flows in grid* method with first and last columns ignored

3.5.3.2 Experimentation and Results

a) Six binary classifications for expressions

The same experimental arrangements were used as for the previous method with a training set consisting of each expression as one set and the remaining five expressions making up the other set. Consequently, the training sets are imbalanced (approximately 1:5, but expressions of sadness and fear of the FG-Net database are much smaller). The test set comprised data derived from the second database and cross validation was performed within each database.

The classifier is SVM, C_SVC, RBF kernel; parameters are selected by 10-fold cross validation on training data. The results are shown in tables 3.11 and 3.12.

Table 3.11: Test results of the FG-NET database for a classifier trained by the MMI database

SVM cross validation on MMI data				
	Group average		Test with FG-NET	
Happiness	Err.R.	1.96%	Err.R.	16.29%
	TPR	92.79%	TPR	60.27%
	TNR	99.53%	TNR	91.43%
Anger	Err.R.	2.61%	Err.R.	24.64%
	TPR	89.45%	TPR	30.73%
	TNR	98.88%	TNR	90.38%
Surprise	Err.R.	2.69%	Err.R.	16.36%
	TPR	87.39%	TPR	47.85%
	TNR	99.30%	TNR	92.91%
Fear	Err.R.	3.05%	Err.R.	10.93%
	TPR	81.03%	TPR	15.93%
	TNR	99.25%	TNR	95.15%
Sadness	Err.R.	2.69%	Err.R.	4.41%
	TPR	84.30%	TPR	55.17%
	TNR	99.17%	TNR	96.40%
Disgust	Err.R.	2.47%	Err.R.	18.33%
	TPR	90.29%	TPR	40.07%
	TNR	99.36%	TNR	91.96%

Table 3.12: Test results of the MMI database for a classifier trained by the FG-NET database

SVM cross validation on FG-NET data				
	Group average		Test with MMI	
Happiness	Err.R.	4.07%	Err.R.	14.60%
	TPR	89.04%	TPR	60.00%
	TNR	98.19%	TNR	92.63%
Anger	Err.R.	5.97%	Err.R.	19.90%
	TPR	88.68%	TPR	49.54%
	TNR	95.82%	TNR	85.85%
Surprise	Err.R.	5.23%	Err.R.	18.88%
	TPR	80.20%	TPR	37.39%
	TNR	98.55%	TNR	89.89%
Fear	Err.R.	2.92%	Err.R.	12.49%
	TPR	76.99%	TPR	18.39%
	TNR	98.75%	TNR	97.51%
Sadness	Err.R.	0.40%	Err.R.	12.13%
	TPR	93.10%	TPR	4.07%
	TNR	99.72%	TNR	99.83%
Disgust	Err.R.	3.60%	Err.R.	19.54%
	TPR	88.36%	TPR	24.10%
	TNR	98.39%	TNR	94.72%

This experiment also used imbalanced training data as in the previous experiment, but as shown in tables 3.11 and 3.12, produced good results on the cross validation process. The rates are all above 80% for the same database cross validation as the best results with all combinations of parameters. This suggests the method is capable of achieving high accuracy when compared with previous methods. When the two databases have been used for training and testing respectively, the results drop sharply, as expected, but are still better than previous feature select methods. Due to the huge variance between the two databases and highly imbalanced dataset, it would be difficult to get good result when test with different databases. The expressions with more data imbalance, sadness and fear, have the worst results.

Increasing the number of cells in the grid

Increasing the number of cells within the grid was investigated as a means of improving classifier performance. The current vector length is 48 (24 grids with x and y velocities). This was increased by starting with 6 cells using a 6 by 6 division of the image as tier 1, then additional tiers of grids are obtained using the following formulae:

Vertical $$T_n.\mathrm{y} = 6 \times (n+1) \times {}^{1}/_{2} \quad (3.33)$$

Horizontal $$T_n.x = T_n.y \times {}^{2}/_{3} \quad ,n = 1,2,3\ldots. \quad (3.34)$$

Where 2/3 represents the effect of removing the two side columns to accommodate the windowing operation (1/6 each, 1/3 total). The same experiments were performed using different grid sizes, from T1 to T5. The grid size and vector length are shown in table 3.13.

Table 3.13: Grids of varying tiers

	T1	T2	T3	T4	T5
Vertical	6	9	12	15	18
Horizontal	4	6	8	10	12
Vector length	48	108	192	300	432

The results (TPR, FPR) of 5 tiers of grids and 6 expressions and 2 databases train and test each other are shown in table 3.14.

Table 3.14: Classifiers trained by grids of varying tiers

			Happiness	Anger	Surprise	Fear	Sadness	Disgust
T1	Test on FG-NET	TPR	60.27%	30.73%	47.85%	15.93%	55.17%	40.07%
		FPR	8.57%	9.62%	7.09%	4.85%	3.60%	8.04%
	Test on MMI	TPR	60.00%	49.54%	37.39%	18.39%	4.07%	24.10%
		FPR	7.37%	14.15%	10.11%	2.49%	0.17%	5.28%
T2	Test on FG-NET	TPR	62.74%	28.84%	48.84%	4.42%	6.90%	22.60%
		FPR	10.11%	4.90%	8.03%	0.44%	1.66%	3.47%
	Test on MMI	TPR	55.08%	42.20%	17.39%	1.72%	0.00%	14.39%
		FPR	2.52%	9.66%	4.27%	0.91%	0.00%	2.37%
T3	Test on FG-NET	TPR	62.19%	29.92%	52.15%	15.93%	24.14%	54.11%
		FPR	8.48%	9.35%	7.52%	3.16%	2.63%	10.84%
	Test on MMI	TPR	67.87%	23.39%	34.78%	12.07%	0.00%	6.12%
		FPR	11.01%	5.69%	7.24%	2.24%	0.00%	0.64%
T4	Test on FG-NET	TPR	61.92%	32.35%	54.13%	8.85%	37.93%	40.41%
		FPR	9.03%	8.98%	8.21%	3.46%	3.60%	10.41%
	Test on MMI	TPR	72.13%	53.67%	35.65%	10.92%	0.00%	28.42%
		FPR	10.07%	15.36%	6.02%	2.08%	0.00%	3.64%
T5	Test on FG-NET	TPR	64.93%	26.15%	48.18%	2.65%	27.59%	44.18%
		FPR	9.39%	7.44%	6.58%	1.76%	3.19%	8.47%
	Test on MMI	TPR	69.84%	45.41%	33.04%	8.62%	0.00%	27.70%
		FPR	7.93%	15.01%	7.32%	1.66%	0.17%	5.46%

Parameters for configuring SVM resulting from the best cross validation and respective classification results are shown in figure 3.15.

Table 3.15: SVM parameters of each experiment

SVM type:	C-SVC		Kernel:	RBF			
			T1	T2	T3	T4	T5
C (10^1)	Happiness	MMI	1.25	6.25	1.25	6.25	1.25
		FG-NET	1.25	1.25	6.25	1.25	1.25
	Anger	MMI	1.25	6.25	1.25	1.25	1.25
		FG-NET	1.25	1.25	0.25	1.25	1.25
	Surprise	MMI	1.25	0.25	1.25	6.25	1.25
		FG-NET	6.25	1.25	6.25	1.25	1.25
	Sadness	MMI	1.25	0.25	1.25	6.25	1.25
		FG-NET	1.25	1.25	1.25	1.25	6.25
	Fear	MMI	1.25	0.25	6.25	1.25	1.25
		FG-NET	6.25	0.25	1.25	1.25	1.25
	Disgust	MMI	1.25	1.25	6.25	31.25	6.25
		FG-NET	1.25	1.25	1.25	1.25	1.25
Gamma (10^-3)	Happiness	MMI	33.75	2.25	2.25	2.25	2.25
		FG-NET	33.75	33.75	2.25	2.25	2.25
	Anger	MMI	33.75	2.25	2.25	2.25	2.25
		FG-NET	33.75	33.75	33.75	2.25	2.25
	Surprise	MMI	33.75	33.75	2.25	2.25	2.25
		FG-NET	33.75	33.75	2.25	2.25	2.25
	Sadness	MMI	33.75	33.75	2.25	2.25	2.25
		FG-NET	33.75	33.75	2.25	2.25	2.25
	Fear	MMI	33.75	33.75	2.25	2.25	2.25
		FG-NET	33.75	33.75	2.25	2.25	2.25
	Disgust	MMI	33.75	33.75	2.25	2.25	2.25
		FG-NET	33.75	33.75	33.75	2.25	2.25

The difference between tiers is not extensive and can be irregular, but the increasing grid size does lead to an increase in vector length and a decrease in the size of each grid window. If the window is too small, then the noise of head movement and personal appearances in the video stream will affect the overall results. Consequently, for future experiments in Chapter 4 on tiers T1 - T3 will be used.

b) Distinguish between pair wise of expressions

A similar experiment to that applied to previous methods, considers evolving classifiers to distinguish between expressions on a pair-wise basis.

Training set and test set: a similar arrangement to previous experiments. Each time has two expressions from 6, there are total 15 (6×5/2) sets of experiments for each database. The numbers of positive and negative results are more balanced except for sadness and fear in the FG-Net database.

Table 3.16: Cross validation of MMI database

N / P		Anger	Surprise	Fear	Sadness	Disgust
Happiness	TPR Ave.	97.7%	97.38%	97.05%	98.03%	98.03%
	TNR Ave.	98.17%	97.83%	96.55%	97.67%	97.12%
Anger		TPR Ave.	99.54%	96.33%	95.87%	96.33%
		TNR Ave.	97.83%	97.7%	94.19%	95.68%
Surprise			TPR Ave.	96.09%	97.39%	98.7%
			TNR Ave.	91.38%	98.94%	98.92%
Fear				TPR Ave.	96.55%	95.4%
				TNR Ave.	97.09%	96.4%
Sadness					TPR Ave.	98.84%
					TNR Ave.	94.96%

Table 3.17: Test results of the FG-NET database trained by the MMI database

N / P		Anger	Surprise	Fear	Sadness	Disgust
Happiness	TPR	87.67%	85.48%	86.03%	92.6%	77.26%
	TNR	82.48%	77.56%	83.19%	100%	82.53%
Anger		TPR	89.22%	78.17%	69.54%	57.68%
		TNR	91.42%	94.69%	65.51%	66.1%
Surprise			TPR	74.92%	89.11%	78.55%
			TNR	37.17%	89.66%	91.44%
Fear				TPR	92.92%	90.27%
				TNR	86.21%	88.01%
Sadness					TPR	68.97%
					TNR	85.96%

This approach performs well on cross validation in same database as shown in table 3.16. When tested on the other database, the sadness and happiness pair has the highest accuracy rate; surprise fear pair, anger sadness pair and anger disgust pair have lower rates as shown in table 3.17.

Table 3.18: Cross validation of FG-NET database

P \ N		Anger	Surprise	Fear	Sadness	Disgust
Happiness	TPR Ave.	95.34%	95.34%	97.26%	100%	96.71%
	TNR Ave.	97.04%	95.71%	91.15%	86.21%	98.97%
Anger		TPR Ave.	97.04%	99.19%	99.46%	95.42%
		TNR Ave.	92.74%	93.81%	93.1%	91.78%
Surprise			TPR Ave.	93.4%	99.34%	96.04%
			TNR Ave.	87.61%	93.1%	98.97%
Fear				TPR Ave.	97.35%	92.92%
				TNR Ave.	100%	99.66%
Sadness					TPR Ave.	93.1%
					TNR Ave.	99.66%

Table 3.19: Test results of the MMI database trained by the FG-NET database

P \ N		Anger	Surprise	Fear	Sadness	Disgust
Happiness	TPR	91.48%	82.95%	91.8%	100%	95.41%
	TNR	86.7%	73.04%	38.5%	19.19%	52.88%
Anger		TPR	90.37%	99.54%	99.54%	61.47%
		TNR	92.17%	43.1%	5.23%	46.04%
Surprise			TPR	83.91%	98.7%	96.96%
			TNR	28.16%	8.72%	65.47%
Fear				TPR	91.38%	40.23%
				TNR	46.51%	90.65%
Sadness					TPR	10.47%
					TNR	99.28%

Similar results are achieved when reversing the databases for training and testing, as shown in table 3.18 and 3.19. Pairs that involve the fear or sadness expression are all disappointing due to the extreme imbalance of training data for these two expressions in FG-Net.

3.6 Summary

Past static analyses of expressions are not reliable due to the differences between people, both in their appearance and their expression. Therefore, a dynamic technique working on the transition of expression has been considered in this chapter. The databases were firstly discussed in the beginning of this chapter because it is the essential condition of the whole project.

In order to analyse the transition of expression dynamically, analysis of optical flow was applied estimating the displacements of every pixel between two frames of video. Two ways to calculate dense optical flow were introduced and the Farneback's fast estimation was used in the thesis. But the dense optical flow contains too much making the vector length too long for the following classification process, so a reduction or selection of the dense optical flow is required.

Three types of features extraction were applied to the dense optical flow data in preparation for subsequent classification. The first approach applied a clustering algorithm, taking the centres of the clusters as feature vectors. This method did not perform well, even when tested with the training video stream, due to the randomness of the clustering algorithm which made the feature vector different each time it was applied to the same flow data. Additionally, the order of the cluster centres was not fixed.

The second approach required picking out the five sub-windows with the highest average flow magnitude, and dividing these sub-windows into nine grids, using the average flow of every grid as a feature vector. This did not perform well in cross validation and the results showed it was greatly affected by the imbalanced training data. This method orders the points from highest flow magnitude sub-window the lowest, but does not include location information. Additionally, the five sub-windows often overlap within one or two areas with high flows.

The third approach segmented the image into a 6 x 6 grid and 36 cells, but ignoring cells in the first and last columns. The average flows velocities of the 24 cells are used as feature vector, resulting in good performance within same database, even when trained with imbalanced data. The method was then evaluated using two different databases for training and testing respectively, and vice versa. The imbalanced dataset still affects classification results, particularly expressions: sadness and fear.

The outcome of the work described in this chapter is a feature extraction method for constructing a vector from optical flow data that is both effective and in a form that can facilitate investigation of genetic programming as a classifier of facial expression.

4 Classification

4.1 Introduction

Chapter 3 introduced a feature extraction method based on dense optical flow and several reduction methods have been discussed. In the end of Chapter 3, results showed the average flow in grids method is better than others. This method is applied in this chapter for comparison of the classifier performances on expression recognition.

This chapter will introduce the CGP based classifier. CGP is very convenient to become a classifier. CGP is used to find the program or computation for given input and output. It can be used to find the network of calculations from the given samples and corresponding class labels. This network of CGP is the classifier for prediction of new input data.

While implementing CGP classifier, many problem need to consider. Such as CGP parameters (mutation rate, number of columns, function set), fitness function, multi-class problem. The CGP parameters will be discussed with a set of 2 dimensional points' classification problems.

Then CGP classifier is used for expression recognition, the feature vector are from previous chapter. Six expressions are recognised by six binary classification processes as in Chapter 3. So the training data is ultra-imbalance. The fitness function should be studied to overcome this problem. The Area under Curve (AUC) is used as the new fitness function.

Overfitting is another problem in classification. Two attempts are made in this chapter to reduce the negative effect: decrease generations of evolution, loose the fitness condition in terminal conditions. Both ways are trying to stop the evolution on CGP classifier earlier than before.

The comparison of expression recognition results is also made between SVM and the CGP classifier. At the end of the chapter, an introduction of the implementation of recognition system is made.

4.2 Cartesian Genetic Programming

CGP has been introduced in Chapter 2. The flexibility and the mechanism makes it can be used in many applications. One of them is to use CGP as a classifier.

4.2.1 CGP Structure

In its original representation, the program compromises a 2 dimensional matrix of cells, each of which is a computational node. Each node can be represented by an n-tuple of integers, representing the *input node numbers* and an index to a *function list* identifying the function defining the computation activity. Every node has a unique number identifier determined by its location in the graph. Input node numbers identify which previous nodes provide the input data for the current node. The function list can be any set of computational functions, such as mathematical equations, circuits and computer programs. The n-tuples for each node form the genotype that represents a CGP or solution computation. From the genotype, a directed connected graph of computation and data can be obtained. There may be a number of nodes in the program that are not connected and therefore have no direct role in the computation.[121] An example of the structure can be found in figure 4.1.

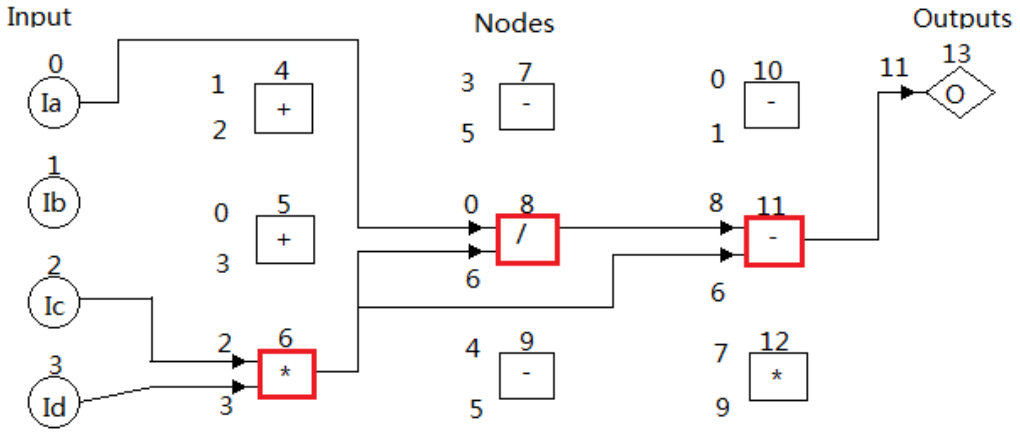


Figure 4.1: Example CGP graph

In the Figure 4.1, circles on the left denoted 'I_x' represent data inputs, rectangles in the middle are nodes and 'O' on the right are outputs. Each of these is uniquely numbered from 0 to 13. The numbers to the left of the boxes are the nodes from which this node obtains its input data. Red nodes are active nodes; those in black are inactive. The arrows show how the computation is directed from input to output. The graph can be coded into a fixed length integer string genotype. Figure 4.2 shows the formation of the program in such a genotype representation.

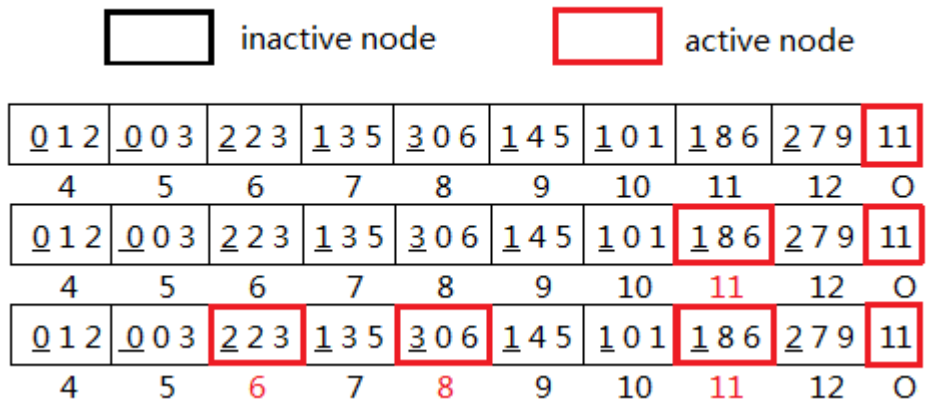


Figure 4.2: Formation from an example CGP graph

A genotype is given and also the number of inputs, number of outputs, rows and columns of nodes are known. Then the active nodes can be found: firstly the entire output nodes must be active; then the nodes connected to output are active; at last all the nodes that link to an active node are active. Once the active nodes are known, each output of these nodes can be calculated. The output of the full computation is then complete.

4.2.2 Genetic Operation

In CGP new individuals are generated using individual (usually the fittest) from the previous generation through mutation. Each number or gene forming the individual or *genome* is subject to mutation at a predetermined *mutation rate*. Should mutation be performed, the value of the *gene* is changed to another random number within the legal range for the gene as shown in figure 4.3.

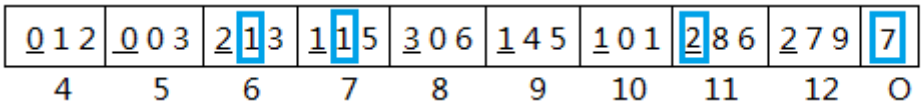


Figure 4.3: Mutation applied to a CGP genome

Depending on its position, the muted gene can be responsible for the input or function of the node. Mutation may also occur at inactive nodes, in which case, the output of the computation will not be affected. However, it is still important to perform these mutations in such nodes as they may be used in future generations. Figure 4.4 shows how the mutation changes the connection and functioning of the program.

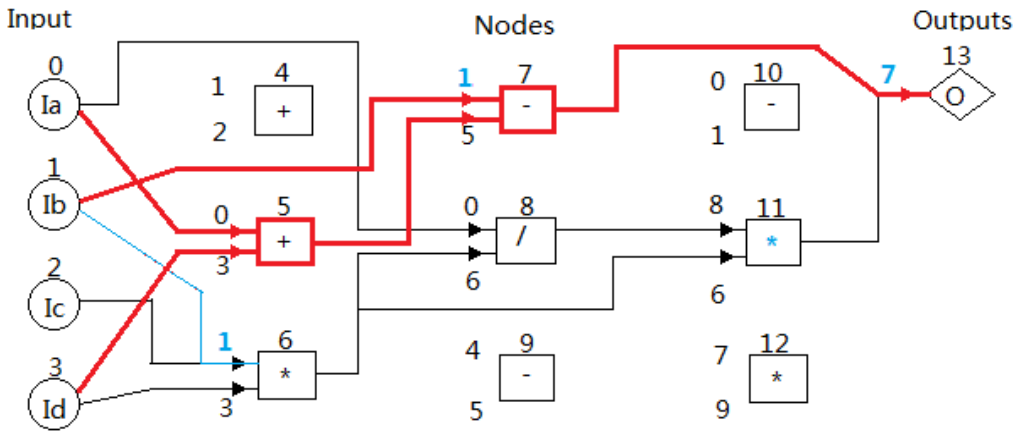


Figure 4.4: The effect of mutation applied to a CGP program

In figure 4.4, red lines and nodes are active the black lines are active connection before mutation. The items in blue are mutated, including the function type, connected node and unconnected node. Anything inactive will not affect the output, so the mutation on those does not show difference in results.

4.2.3 Evolution Strategy

A 1+λ evolution strategy is widely used for CGP, where λ is usually chosen to be 4. This means the population size is 5, and one, the fittest individual is selected and copied or *cloned* to generate 5 new individuals, 4 of which are subject to mutation, as the next generation.[121]

- Initialisation: generate 5 individual solutions randomly.
- Evaluate fitness of all individuals: because CGP is to find the computation with known input and output. The fitness is usually the error between CGP output and desired output. The lower is better.
- Select fittest individual as the parent.
- If this parent's fitness satisfied, then Break.
- Otherwise generate new individuals by mutating from the parent
- Repeat the process until maximum generations reached.

4.3 CGP classifier

The CGP is finding the connection of operations that fills the black box between the known input and output.

For a classifier, the vector for classifying is the input and then it returns the class label as output in the testing phase. For the training phase, the input and its class label, which should be the desired output, is known. Then the classifier learns all the training data and develops itself to find the best internal parameters for the relationship between input and output. This procedure leads classifier divides input samples into their categories. After the classifier is well trained, in another word best parameters are found, usually a file with full internal setting is generated. With these well-trained setting, the classifier can predict the output from an unknown input sample. The solution that CGP works out can be such relationship from input to output. Once CGP finds the best solution of relationship between all the samples and their labels, this specific connection of CGP network will be used for prediction of which class a new input sample belongs.

4.3.1 Classifier Algorithm

4.3.1.1 Input and Output

For a CGP classifier, the input is samples in the form of a fixed length vector. For one classification problem, the length of the input should be fixed, which is same as other classifiers. Each sample has a label indicated the class it belongs. The label is usually integer number, like 1, 2 and/or more depends how many classes there are. If the number of classes is two, also known as binary classification problem, then a threshold on output can be set to divide the two classes. That is, if the output is above the threshold, then the input sample belongs to one class, vice versa.

In the training phase, two matrixes are required: a matrix of training data and a one column matrix with corresponding class labels. One row in training data matrix is one sample vector. The number of the matrix rows is the number of samples. The number of matrix columns is the length of sample vector, and the length is fixed. The number of label matrix rows is equal to the row number of training matrix, and the labels are corresponding to the vectors that are at the same rows.

4.3.1.2 Fitness Function

A major part in the evolution process is evaluating the fitness of the solution. One solution is a specific connection between CGP nodes, and with this network CGP can predict class labels of samples. Fitness of one solution should indicate how well the solution can classify the training data. The method to evaluate the fitness, fitness function, should be simple and straight forward. One fitness function could be count the number of correct prediction on training data.

4.3.1.3 Procedure of CGP Classifier

The process is almost same as standard CGP. In the "evaluating fitness of individual solution" step, each row of the training data matrix is compared the CGP output from this chromosome and its true label from label matrix. For every chromosome, all the training samples should be examined the output and the desired class label. Then the number of correct classification is the fitness value of this chromosome. After generations, the fitness value is supposed to grow, means the solution gets better and can classify more training samples correctly.

4.3.2 Experiments

In this section, a set of experiments are introduced based on CGP classifier. The samples are two dimensional data (points on 2D coordinate). The classes or segmentations are decided by one or more functions.

4.3.2.1 Simple Sinusoidal Wave

The test data is randomly generated points in 100 by 100 (10,000 total) data space. The classes are divided by two functions:

$y < 10*\sin(x/10) + 50$ class 1

else $x < 50$ class 2

else ($x > 50$) class 3 (4.1)

The outputs of CGP have the even threshold for three classes (1/3 of output range).

The samples are shown in the figure. Different colours mean three classes.

a) 1000 points for training

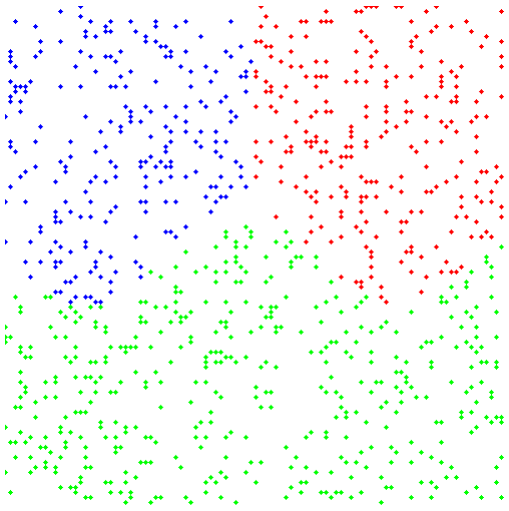


Figure 4.5: 1000 training samples of simple sinusoidal wave segmentation

Training data

Firstly, 1000 samples are generated and they take 10% of the whole data space as the training data. From the figure above, the points are quite dense and it is not difficult to find the boundary between classes.

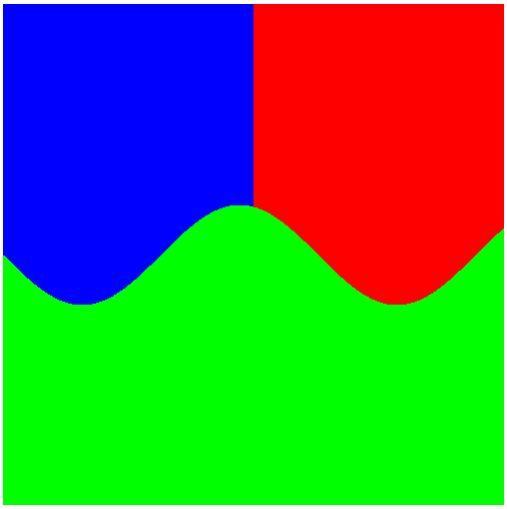


Figure 4.6: Ground truth of simple sinusoidal wave segmentation

Ground truth

The two functions divide the three classes, a sinusoidal wave and a straight line. It is a simple classification problem.

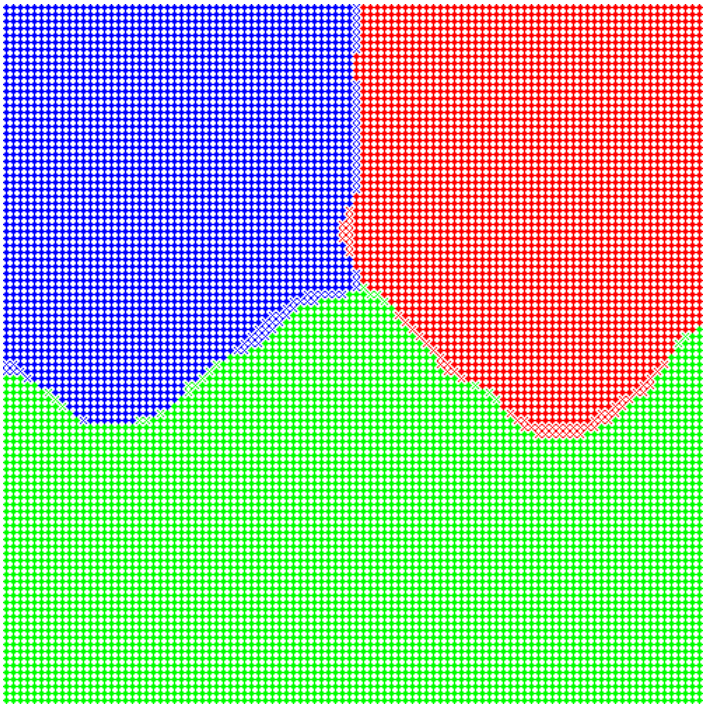
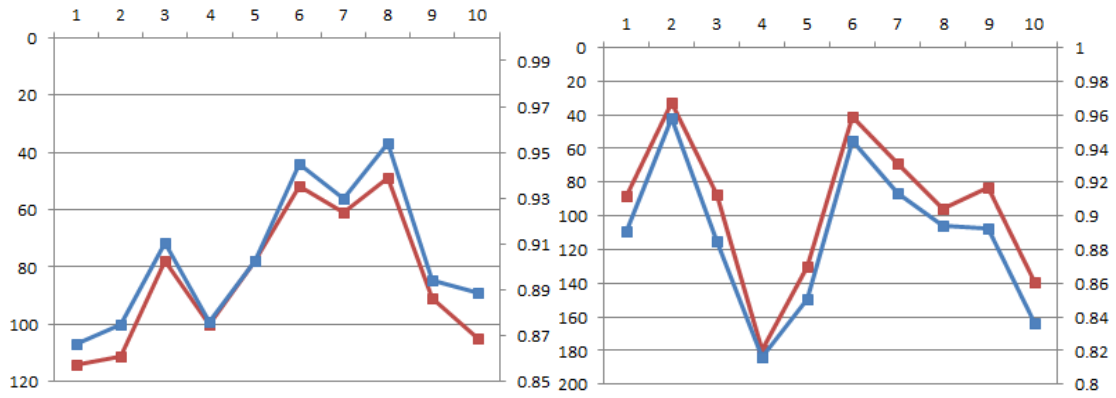


Figure 4.7: SVM result. Accuracy of the whole space (100*100) is 98.57%

The classification result by SVM is shown in the figure. The three colours are the predictions from the SVM classifier. The solid dots (darker colour) mean the correct recognitions and the crosses (lighter colour) mean the misclassifications. The segmentation is very close to the ground truth. SVM performs always better than CGP classifier in the following segmentation problems. This is because the nature advantage of SVM on segmentation problems.

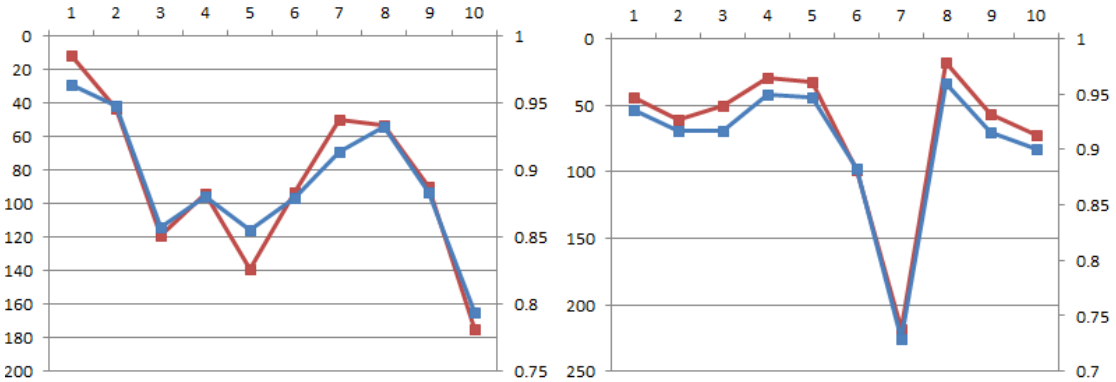
CGP results

All the CGP classifier experiments run 100,000 generations. Other parameters: columns 200 or 1000, mutation rates 0.1 or 0.008. The evolution process takes from 2 minutes(less columns, less samples) to 20 minutes. The computation time of CGP classifier costs much more than SVM. The CGP algorithm depends on heavy randomness, so the results are different when repeat the same experiment. Every experiment runs 10 times, then the fitness of training data and the accuracy of the whole data space are compared.



CGP: 200 columns, 0.1 mutation rates

1000 columns, 0.1 mutation rates



200 columns, 0.008 mutation rates

1000 columns, 0.008 mutation rates

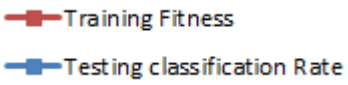


Figure 4.8: Plot of fitness and accuracy in ten runs of simple sinusoidal wave segmentation (1000 training points)

Figures 4.8 are the relations of training fitness and accuracies among ten runs. The horizontal axis has ten points and they stand for the ten runs of the CGP procedure. The left vertical axis is the final fitness on training data, the smaller is better. The right vertical axis is the accuracy on the whole data space (100 * 100 points), the larger is better. Four figures are each combination of two mutation rates and two columns.

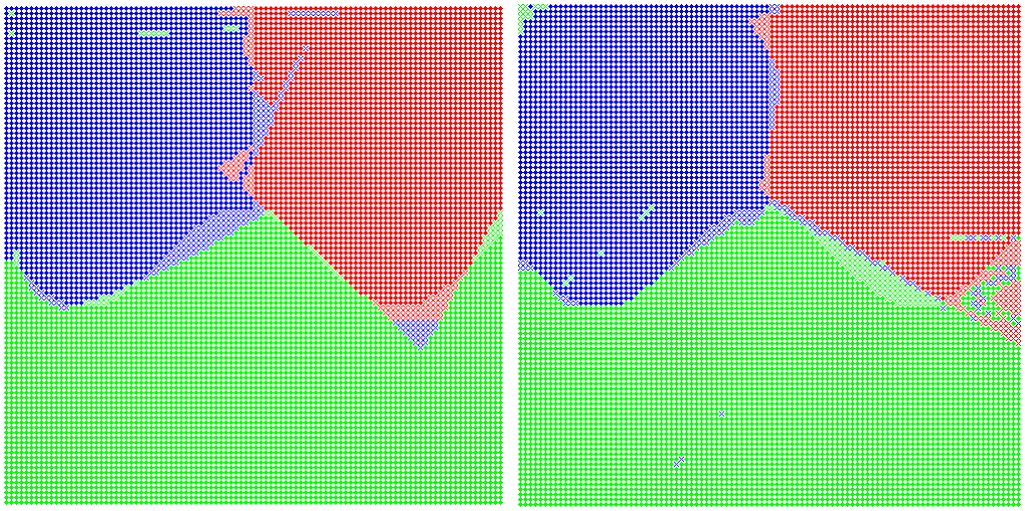
From the figures, the line of accuracies follows the training fitness nearly perfect. The axis scales of fitness values and accuracies are different among figures. As the scale changing, the distance of the two lines will be apart. But the shapes of the lines, which connect the 10 points and show the relations of points, have similarities. In another word, the fitness function, which counts the correct classification of training data, can evaluate the classification performance on new data.

The comparison between four combinations of CGP parameters is in the following table.

Table 4.1: Performance of different CGP parameters on simple sinusoidal wave segmentation (1000 training points)

Samples, Mutation R.	Accuracy			Fitness		
	Ave.	Max.	Min.	Ave.	Max.	Min.
200,0.1	0.90419	0.9539	0.8663	83.9	114	49
1000,0.1	0.88809	0.9579	0.8164	94.6	180	33
200,0.008	0.89066	0.9636	0.7941	86.8	175	12
1000,0.008	0.90539	0.9599	0.7295	68.2	218	18

The performances of the four combinations are very close. The result of 1000 columns and 0.008 mutation rates is slightly better than others on the average of recognition rates and fitness value. The individual of highest accuracy is obtained by the parameter of 200 columns and 0.008 mutation rates, and is shown in the following figure (accuracy is 96.36%).



200 columns, 0.008 mutation rates (96.36% accuracy)

1000 columns, 0.008 mutation rates (95.99% accuracy)

Figure 4.9: CGP results of the simple sinusoidal wave segmentation (1000 training points)

b) 200 samples for training

In the previous experiment, the training data set is very large which takes 10% of the whole data space. Then the next experiment is reducing the number of training samples to 200, which is 2% of the whole data space. The fewer training data makes the classification more difficult.

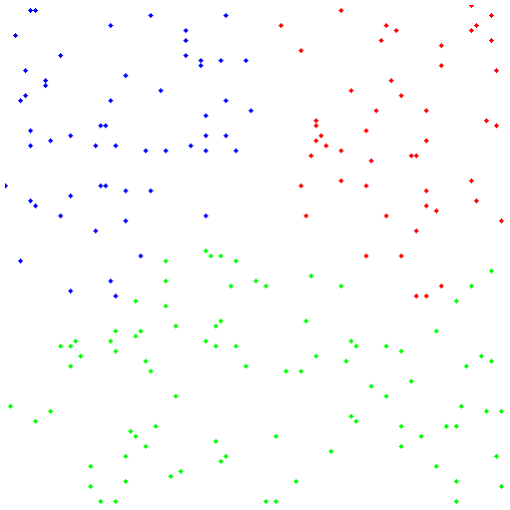


Figure 4.10: Training samples (200 points) for simple sinusoidal wave segmentation

Figure 4.10 are 200 points of training samples. Three colours mean different classes.

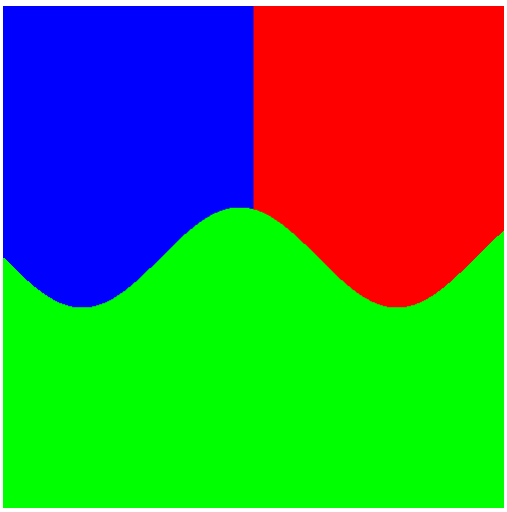


Figure 4.11: Ground truth of, same as the previous experiment

The training data are randomly selected from the ground truth of the data space. The ground truth will not change as training data.

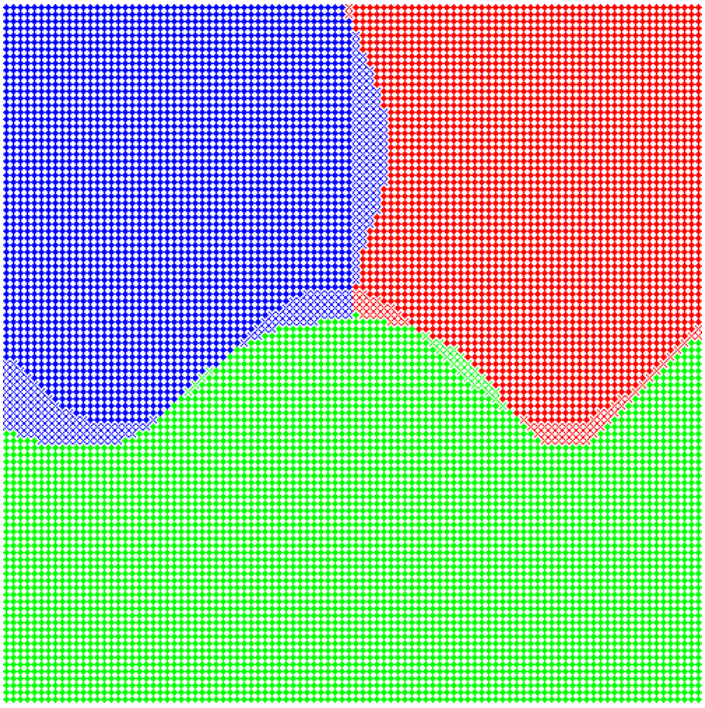
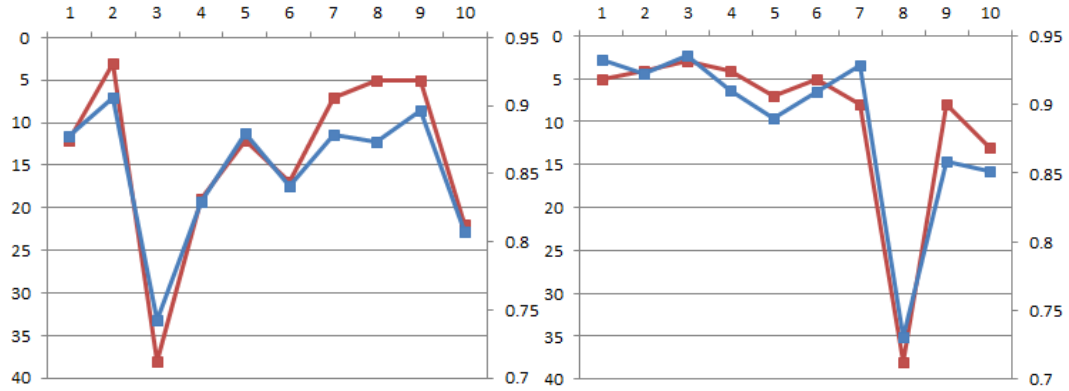


Figure 4.12: SVM result of simple sinusoidal wave segmentation with 200 training points (96.23%)

From figure 4.12, cross (lighter colour) is misclassification, solid dot (darker colour) is correct recognition. SVM is very strong to this kind of problem. The SVM result is still close to the ground truth.

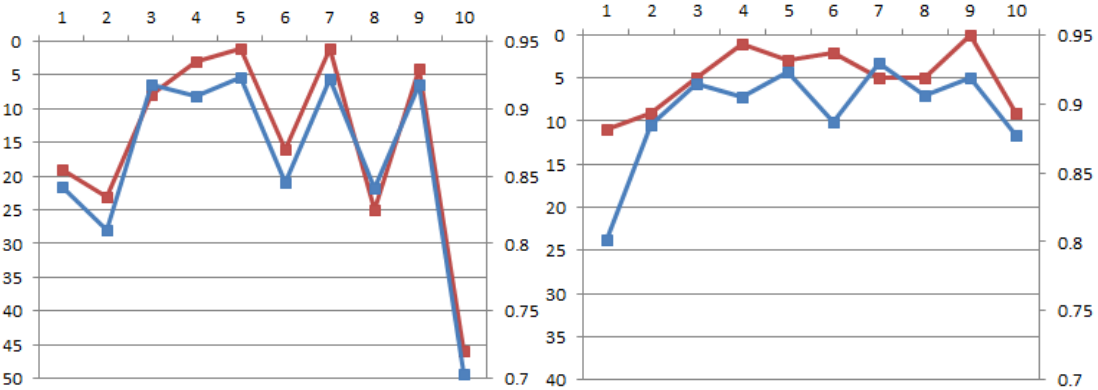
CGP results:

Ten runs of CGP classifier training are made. The following figure shows the relations between final fitness value of training process and the accuracies on testing data.



CGP: 200 columns, 0.1 mutation rates

1000 columns, 0.1 mutation rates



200 columns, 0.008 mutation rates

1000 columns, 0.008 mutation rates

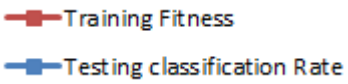


Figure 4.13: Plot of fitness and accuracy in ten runs on simple sinusoidal wave segmentation (200 training points)

The horizontal axis is the run number of CGP. The left vertical axis is the final fitness on training data, the smaller is better. The right vertical axis is the accuracy on the whole data space (100 * 100 points), the larger is better. Four figures are each combination of two mutation rates and two columns.

From figures 4.13, the accuracies generally follow the training fitness. In the figure of 1000 columns and 0.008 mutation rates, the fitness is very close (7 of 10 runs have less than 5 fitness value differences), and two lines do not follow each other very well. The conflict is on the 4th and 6th runs, the fitness values are high among 10 points, but the accuracies are low in 10 percentage points.

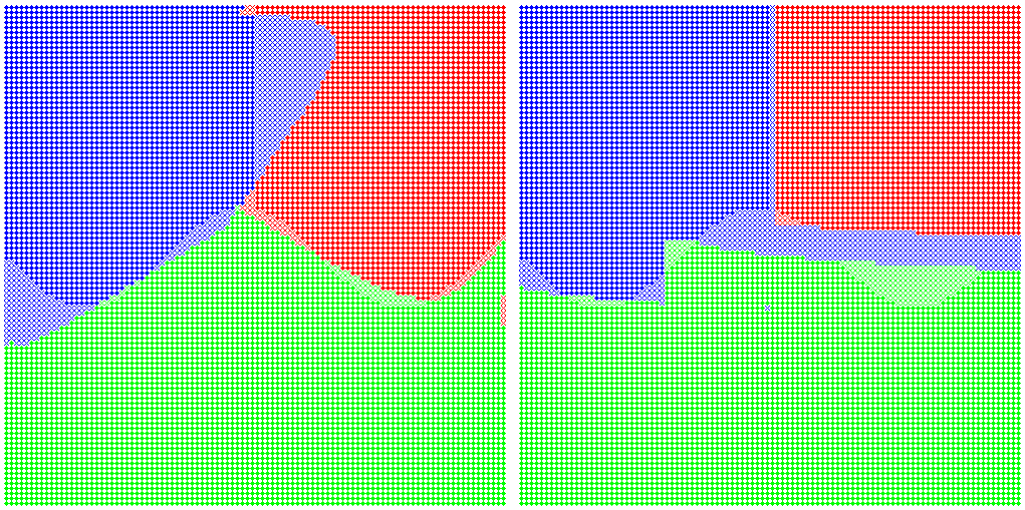
The comparison between the four combinations of CGP parameters are shown in the following table.

Table 4.2: Performance of different CGP parameters on simple sinusoidal wave segmentation (200 training points)

Samples, Mutation R.	Accuracy			Fitness		
	Ave.	Max.	Min.	Ave.	Max.	Min.
200,0.1	0.85325	0.9059	0.7423	14	38	3
1000,0.1	0.88698	0.9361	0.731	9.5	38	3
200,0.008	0.86323	0.9231	0.7033	14.6	46	1
1000,0.008	0.89472	0.9293	0.8018	5	11	0

From the averages of the accuracies, the results of 1000 columns are better than 200 columns. And the smaller mutation rates have better average performance with same columns. The 1000 columns and 0.008 mutation rates parameters has the best average performance, just a slightly lower maximum accuracy than parameters of 1000 columns and 0.1 mutation rates.

Best accuracy of 93.61% is achieved by the third run with 1000 columns and 0.1 mutation rates. The second best is by parameter of 1000 columns and 0.008 mutation rates. The result of this classifier is shown in the following figure.



1000 columns, 0.1 mutation rates

93.61% accuracy

1000 columns, 0.008 mutation rates

92.93% accuracy

Figure 4.14: CGP result on 200 training data simple sinusoidal wave segmentation experiment

4.3.2.2 Third Order Polynomial

The next experiment is a more complex region segmentation problem on a 2D space. The function is:

$$z = x^2 - 3x + y^3 - 3y \qquad x, y \in [-2,2], z \in [-4,4] \qquad (4.2)$$

Classes are divided by the following threshold:

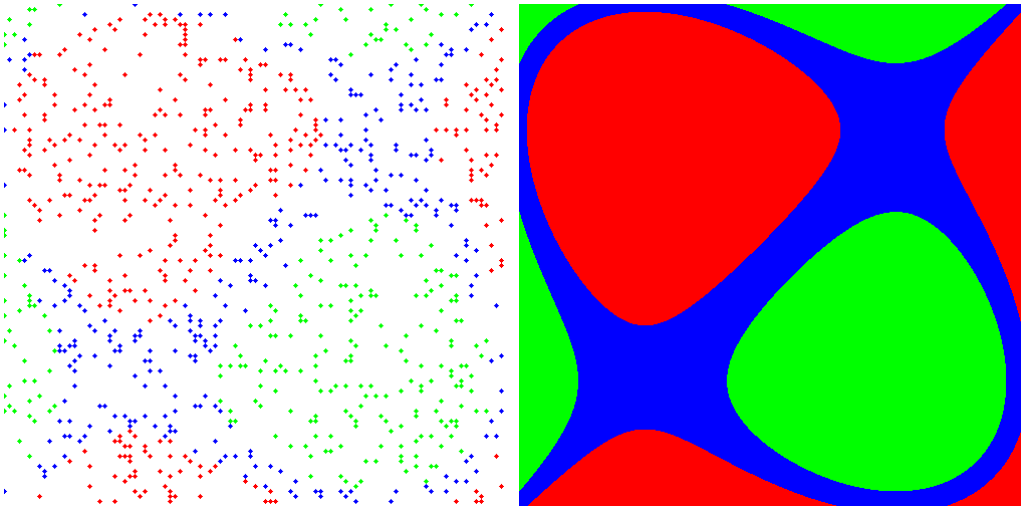
Class 1: $z \in [-4, -1)$

Class 2: $z \in [-1,0.5)$

Class 3: $z \in [0.5,4]$ (4.3)

The data space is still 100 by 100, but scale the x and y to the range in the equation before processing. The experiments are the same as the previous one, first train with 1000 samples (10% of data space) then 200 samples (2%). The parameters are the same combination as previous experiment.

a) 1000 samples for training



(a)Training data (b)Ground truth

Figure 4.15: (a)Training data (1000 points) and (b) ground truth of segmentation based on third order polynomial

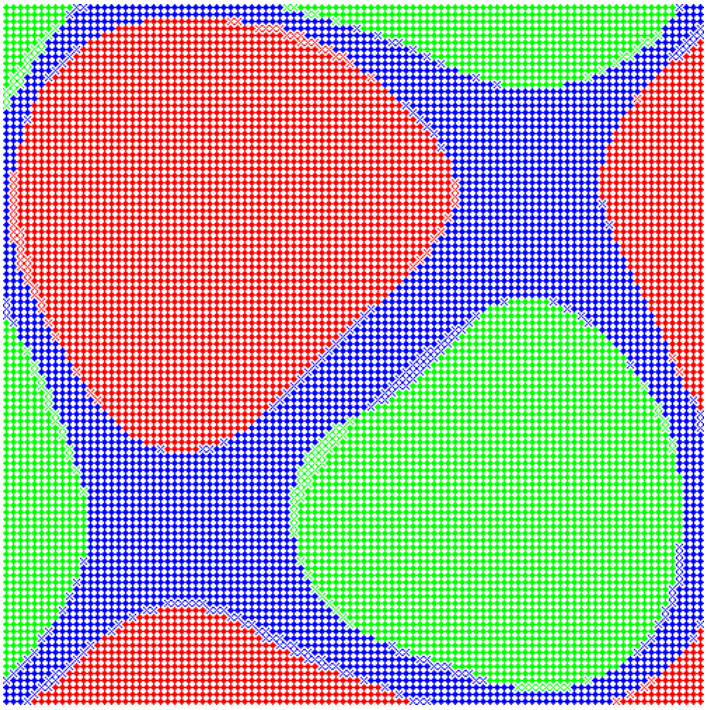


Figure 4.16: SVM result: 96.34% accuracy of segmentation recognition based on third order polynomial (1000 training points)

CGP results

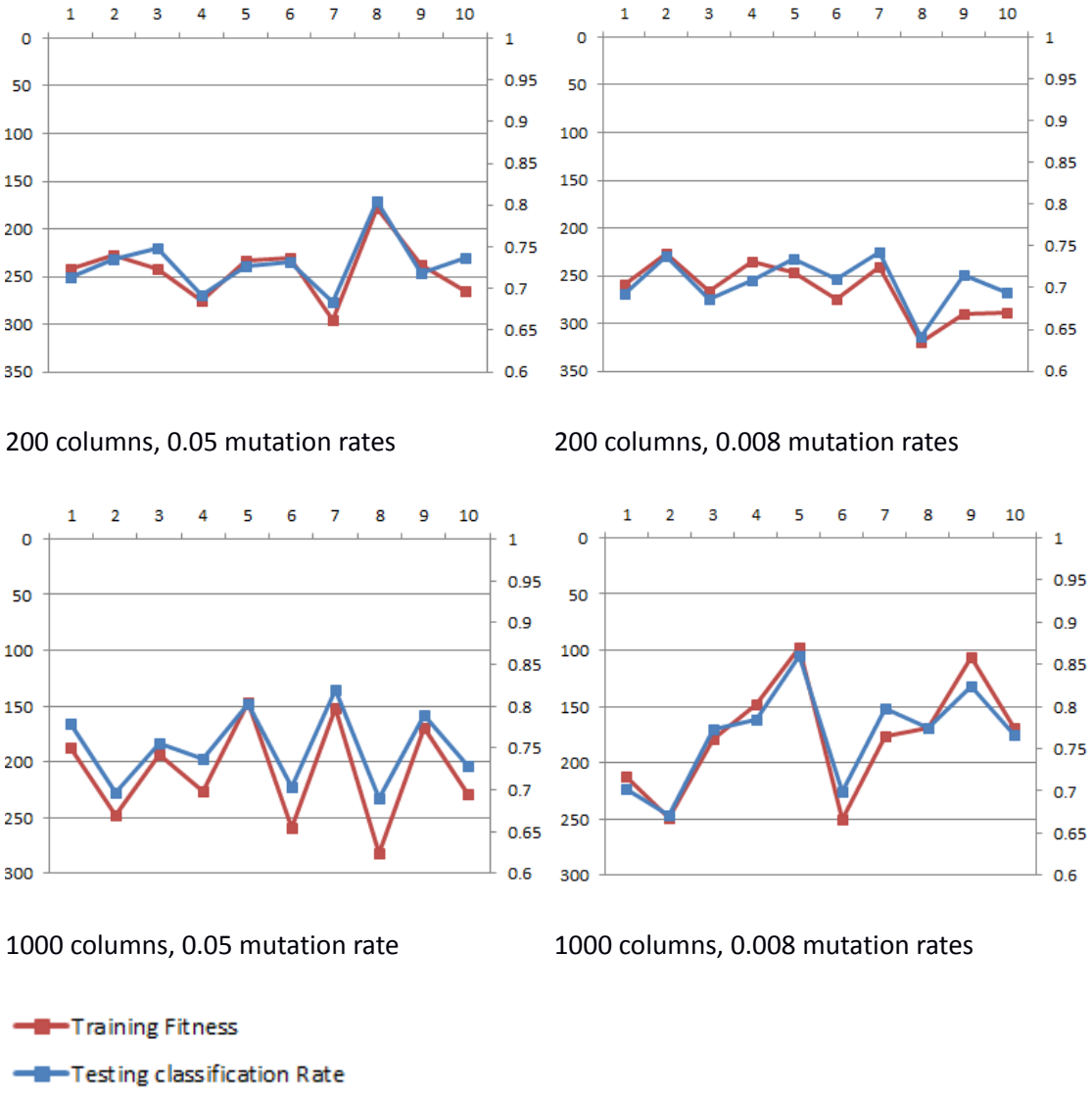


Figure 4.17: Plot of fitness and accuracy of segmentation problem based on a third order polynomial equation (1000 training points)

Ten points on the horizontal axis mean ten runs of CGP classifier. The left vertical axis is the final fitness on training data, the smaller is better. The right vertical axis is the accuracy on the whole data space (100 * 100 points), the larger is better. Four figures are each combination of two mutation rates and two columns.

Table 4.3: Performance of different CGP parameters on the third order polynomial equation based segmentation problem (1000 training points)

Samples, Mutation R.	Accuracy			Fitness		
	Ave.	Max.	Min.	Ave.	Max.	Min.
200,0.1	0.7294	0.8054	0.6843	242.7	296	178
1000,0.1	0.7065	0.7431	0.6421	264.7	319	227
200,0.008	0.75006	0.8197	0.6896	209.3	282	147
1000,0.008	0.76521	0.8605	0.6706	175.5	250	97

The result with 1000 columns, 0.008 mutation rates performs better among other CGP classifier parameters. The smaller mutation rates have better performance when the column number is the same.

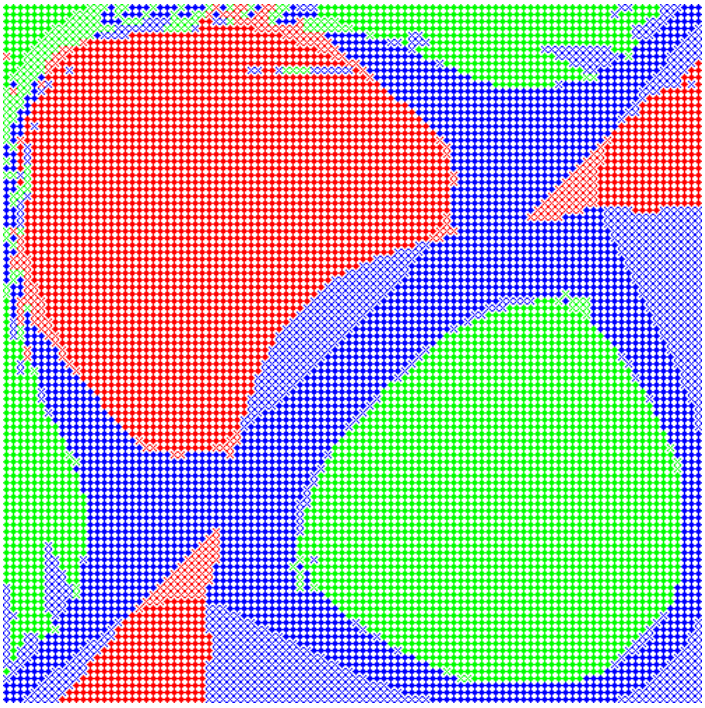
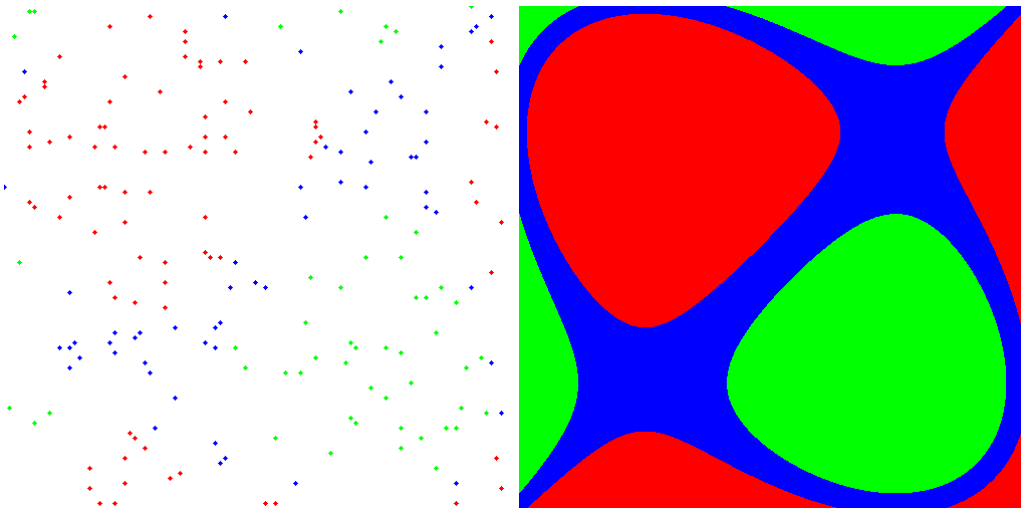


Figure 4.18: Best CGP classifier result (86.05%, 1000 columns and 0.008 mutation rate) on third order polynomial equation based segmentation recognition (1000 training points)

b) 200 samples for training



(a)Training data (b)Ground truth, same as 1000 samples

Figure 4.19: (a)Training data (200 points) and (b) ground truth of segmentation based on third order polynomial

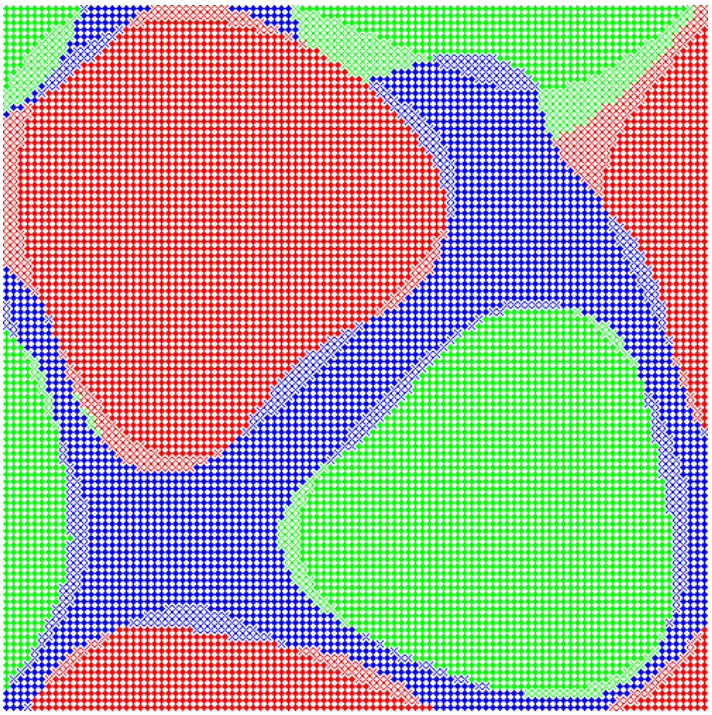


Figure 4.20: SVM result is at 88.6% accuracy on the third order polynomial equation based segmentation problem (200 training points)

In figure 4.20, the three colours mean the class prediction given by SVM, the cross marks (lighter colour) are the misclassifications, solid dots (darker colour) are correct recognition.

CGP classifier performance

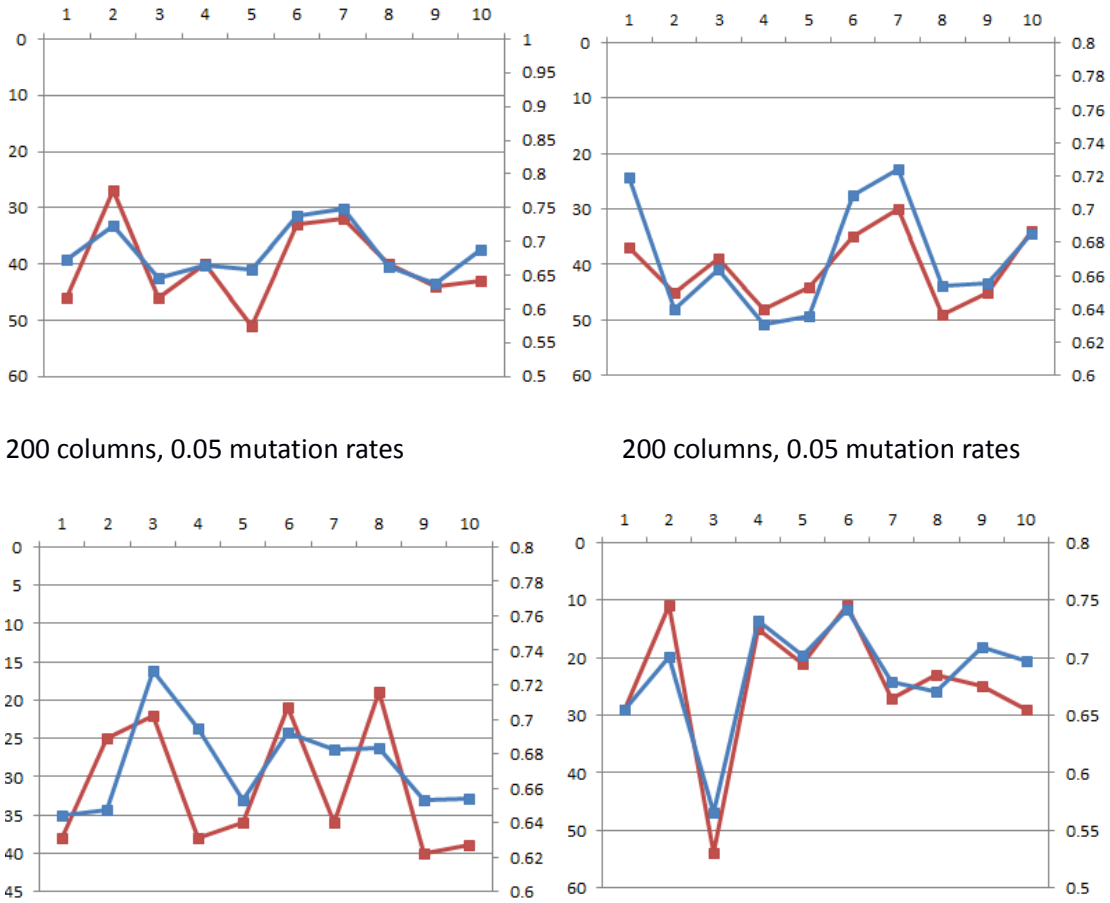


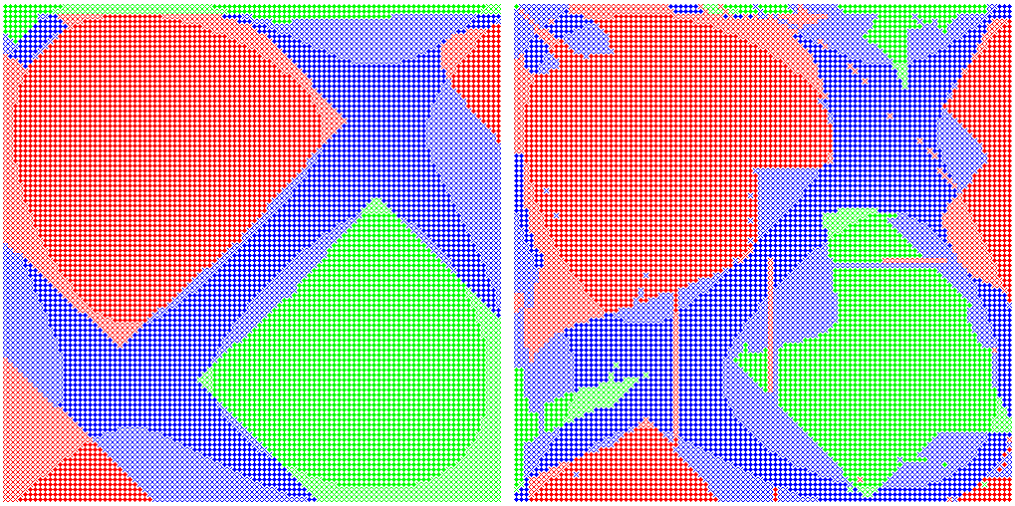
Figure 4.21: Plot of fitness and accuracy of segmentation problem based on a third order polynomial equation (200 training points)

Ten points on the horizontal axis mean ten runs of CGP classifier. The left vertical axis is the final fitness on training data, the smaller is better. The right vertical axis is the accuracy on the whole data space (100 * 100 points), the larger is better. Four figures are each combination of two mutation rates and two columns.

Table 4.4: Performance of different CGP parameters on the third order polynomial equation based segmentation problem (200 training points)

Samples, Mutation R.	Accuracy			Fitness		
	Ave.	Max.	Min.	Ave.	Max.	Min.
200,0.1	0.68436	0.7489	0.6384	40.2	51	27
1000,0.1	0.6715	0.7241	0.6304	40.6	49	30
200,0.008	0.67329	0.7281	0.6442	31.4	40	19
1000,0.008	0.68548	0.7418	0.5657	24.5	54	11

In the table 4.4, the fitness values of smaller mutation rates are better than others with same column number. But the differences of accuracies between them are very small.



Best CGP classifier result: 200 C, 0.1 MR (74.89%) 1000 C, 0.008 MR (74.18%)

Figure 4.22: CGP results on third order polynomial equation based segmentation recognition (200 training points)

4.3.2.3 Complex Sinusoidal Wave

Sinusoidal function:

$$z = \sin(4xy) \qquad x, y \in [-1,1], z \in [-1,1] \qquad (4.4)$$

Classes are divided by the thresholds:

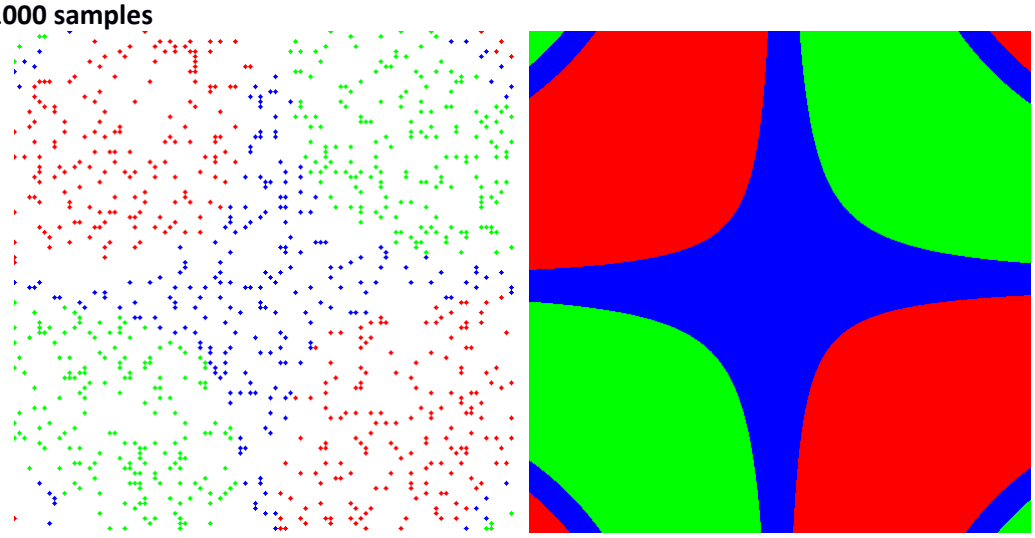
Class 1: $z \in [-1, -0.7)$

Class 2: $z \in [-0.7, 0.2)$

Class 3: $z \in [0.2, 1]$ (4.5)

Other settings are same as previous experiment.

a) 1000 samples



(a)Training data (b)Ground truth

Figure 4.23: (a)Training data (1000 points) and (b) ground truth of segmentation based on complex sinusoidal wave

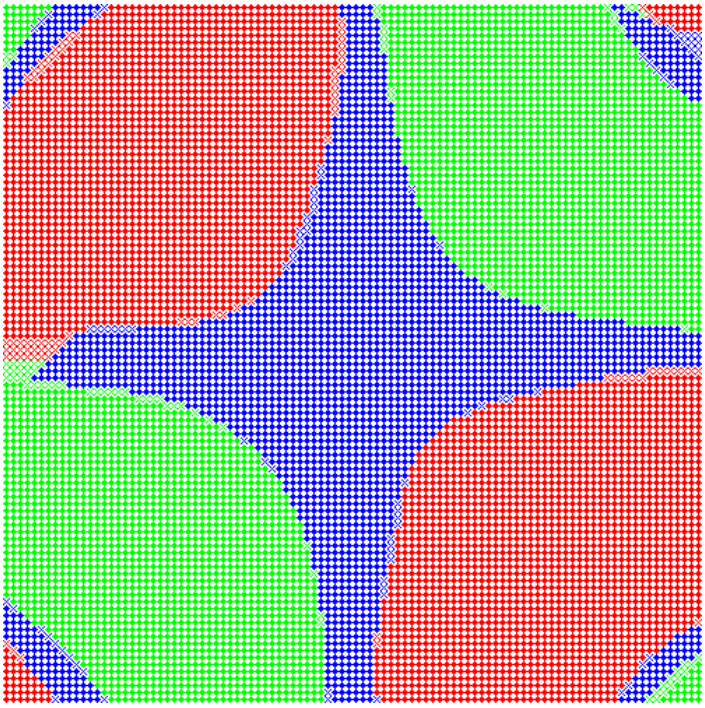


Figure 4.24: SVM result 97.55% accuracy of complex sinusoidal wave based segmentation recognition (1000 training points)

CGP classifier performance

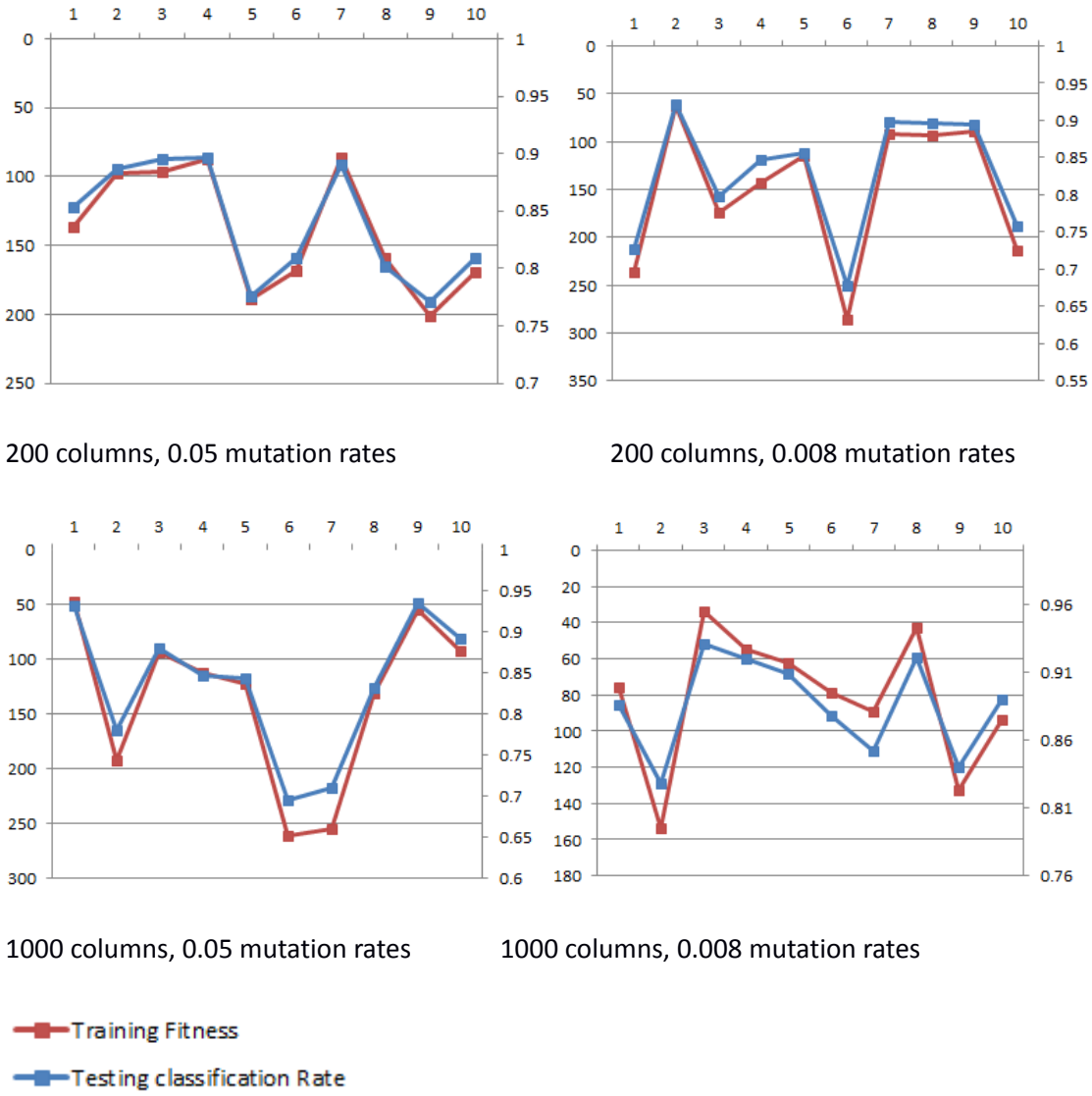


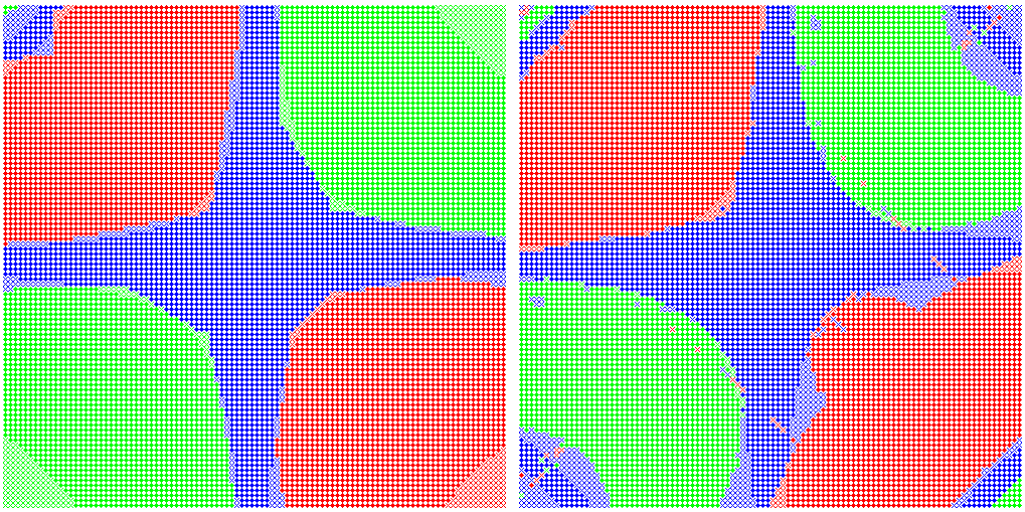
Figure 4.25: Plot of fitness and accuracy in ten runs of complex sinusoidal wave segmentation (1000 training points)

Ten points on the horizontal axis mean ten runs of CGP classifier. The left vertical axis is the final fitness on training data, the smaller is better. The right vertical axis is the accuracy on the whole data space (100 * 100 points), the larger is better. Four figures are each combination of two mutation rates and two columns.

Table 4.5: Performance of different CGP parameters on complex sinusoidal wave segmentation (1000 training points)

Samples, Mutation R.	Accuracy			Fitness		
	Ave.	Max.	Min.	Ave.	Max.	Min.
200,0.1	0.83905	0.8971	0.7709	138.9	201	86
1000,0.1	0.82725	0.9218	0.6784	150.7	286	63
200,0.008	0.83464	0.9349	0.6952	136.3	261	47
1000,0.008	0.88551	0.9307	0.8277	81.9	154	34

The classifier with 1000 columns and 0.008 mutation rates has the best average performance. The results of first and second high accuracy are shown in the figure below. They are with the 200 columns, 0.008 mutation rates and 1000 columns, 0.008 mutation rates.

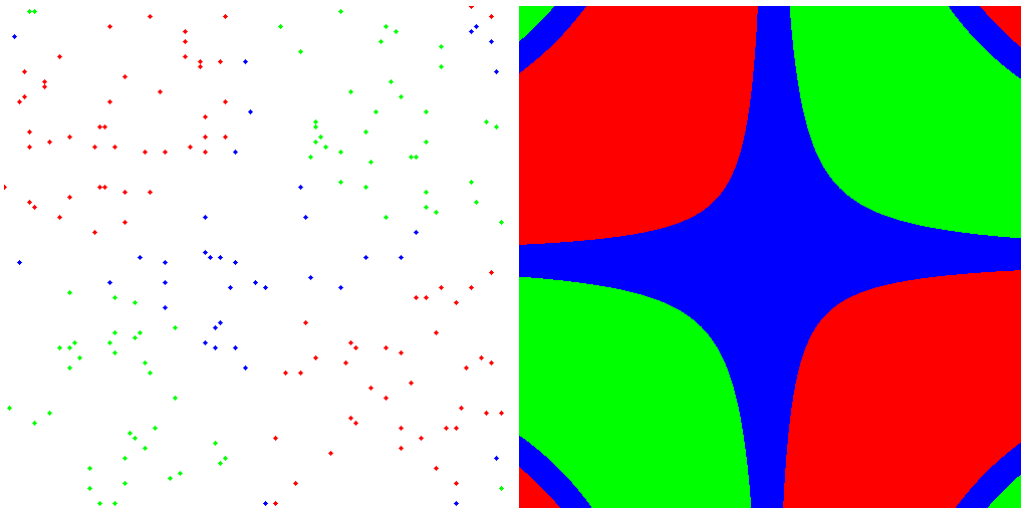


Best CGP result: 200 C, 0.008MR (93.49%) 1000 C, 0.008MR (93.07%)

Figure 4.26: CGP results of the complex sinusoidal wave segmentation (1000 training points)

In figure 4.26, the results are above 90% accuracy. The difference between the two is that the segmentation result with 200 columns has simpler line type than the 1000 columns one.

b) 200 samples



(a)Training samples (b)Ground truth, same as 1000 samples

Figure 4.27: (a)Training data (200 points) and (b) ground truth of segmentation based on complex sinusoidal wave

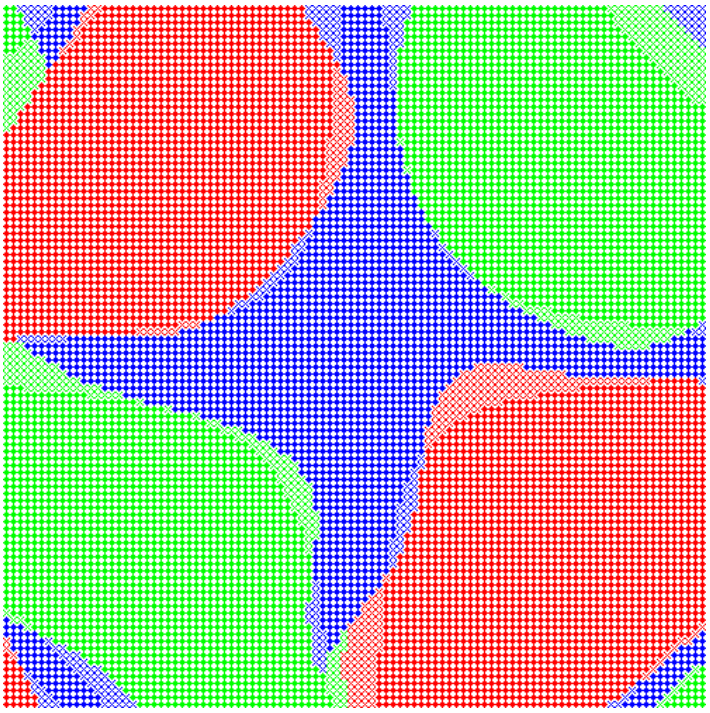
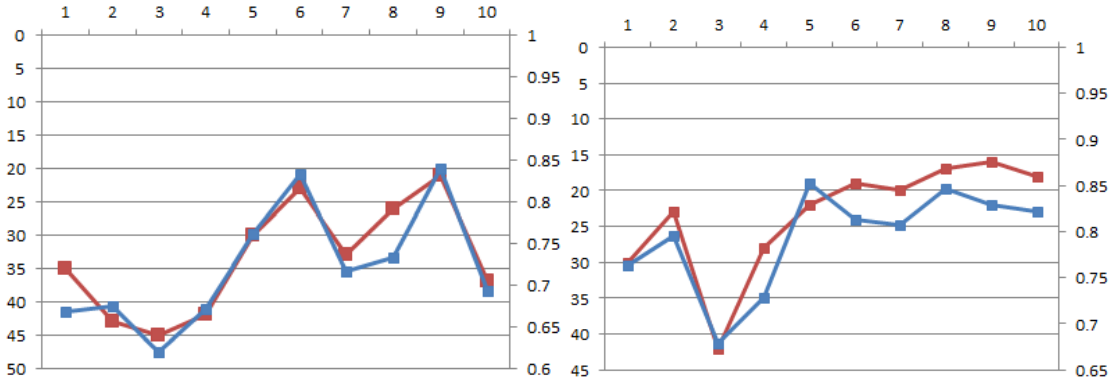


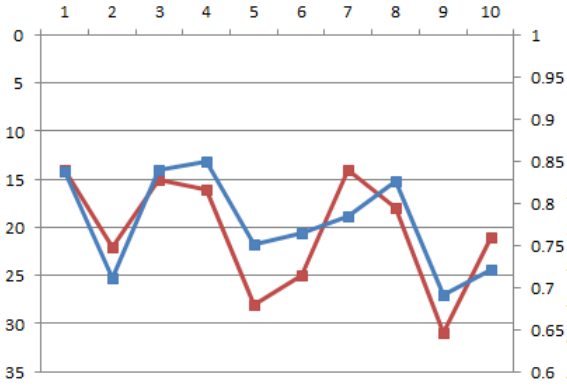
Figure 4.28: SVM result 91.73% accuracy of complex sinusoidal wave based segmentation recognition (200 training points)

CGP performance

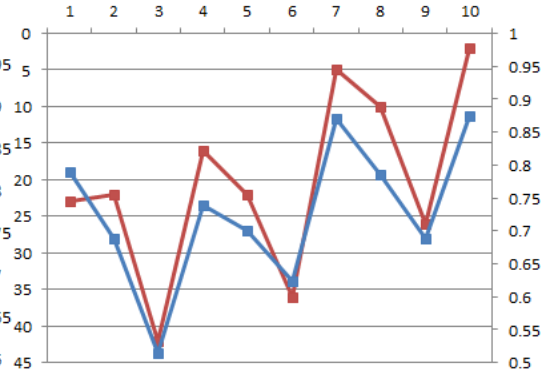


200 columns, 0.05 mutation rates

200 columns, 0.008 mutation rates



1000 columns, 0.05 mutation rates



1000 columns, 0.008 mutation rates

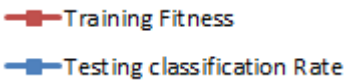


Figure 4.29: Plot of fitness and accuracy in ten runs of complex sinusoidal wave segmentation (200 training points)

Ten points on the horizontal axis mean ten runs of CGP classifier. The left vertical axis is the final fitness on training data, the smaller is better. The right vertical axis is the accuracy on the whole data space (100 * 100 points), the larger is better. Four figures are each combination of two mutation rates and two columns.

Table 4.6: Performance of different CGP parameters on complex sinusoidal wave segmentation (200 training points)

Samples, Mutation R.	Accuracy			Fitness		
	Ave.	Max.	Min.	Ave.	Max.	Min.
200,0.1	0.726656	0.8395	0.6193	33.33333	45	21
1000,0.1	0.797011	0.8523	0.6783	22.77778	42	16
200,0.008	0.771356	0.8493	0.6909	21.11111	31	14
1000,0.008	0.720567	0.8752	0.514	20.11111	42	2

The 0.008 mutation rates experiments have better fitness value and maximum accuracy. But the average accuracy does not in this experiment. The best CGP classifier result is shown below.

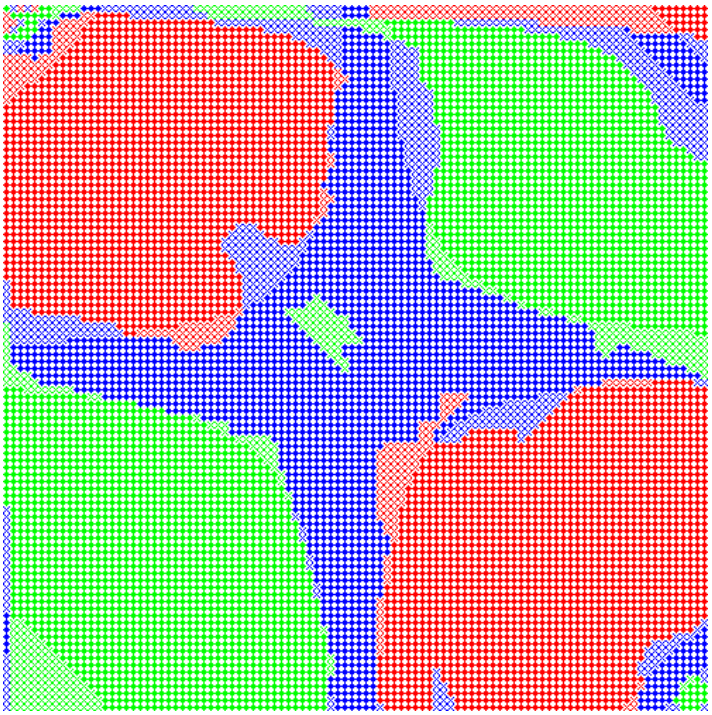


Figure 4.30: 1000 columns, 0.008 mutation rates (87.52% accuracy) on complex sinusoidal wave segmentation (200 training points)

Figure 4.30 is the best accuracy on the complex sinusoidal wave segmentation problem with 200 training points. The classifier has the setting of 1000 columns and 0.008 mutation rates and achieves 87.52% accuracy. The cross mark (lighter colour) means on that point the classifier gives the wrong prediction; the solid dot (darker colour) means the correct recognition.

4.3.3 Conclusion

This fitness function, counting the correct classifications of each individual solution, cannot only evaluate the performance on training data, but also reflect the classifier performance on new data or the whole data space. From the figures in this section, the overall accuracy is nearly always following the wave of training fitness values. By using the fitness function of counting the correct classifications, CGP classifier can achieve good classification performance. But the experiments show the SVM is stronger than CGP on the segmentation problem. SVM is finding the boundary between classes that has the largest distances to the classes. It is suitable for this kind of problems in theory.

The CGP evolution process is highly random numbers based. The result of one experiment can be very different when repeating with different random seeds. So each experiment is done 10 times and comparison is made on average, maximum and minimum fitness values and accuracies in order to get the general effect of the parameters and avoid the effects of randomness. From the results in this section, larger columns and smaller mutation rates usually have the better fitness value and are suggested having better performance according to the relation between fitness value and classification performance. The parameter of 1000 columns and 0.008 mutation rate has the best average accuracy in five of all six experiments and the best average fitness value in all experiments. This setting is also suggested in Julian Miller's book[121].

4.4 Application to Optical Flow

The CGP classifier, with the fitness function of counting correct classification, is applied to following experiments for expression recognition.

The training and testing data are from two databases: MMI and FG-NET. The video clips are cropped manually that only the face area remains. The rising parts of expressions are cut and dense optical flows are extracted between one frame and the fifth after. Each whole picture's dense optical flow is one sample and the total numbers of samples are in table 3.1 and 3.2. Then the flow data are reduced by averaging the grids, the image is divided into 24 equal grids. There are six binary classification experiments for six expressions. Each experiment requires the positive samples, the samples of corresponding expression, and the negative samples, samples of all other five expressions. The numbers of samples for experiments are shown in table 3.8. Every experiment is done twice: training with data from MMI and testing with data from FG-NET; then training with data from FG-NET and testing with data from MMI database.

4.4.1 Function Set

$$x$$

$$y$$

$$\sqrt{x+y}$$

$$|x+y|(mod\ 255)$$

$$|x-y|(mod\ 255)$$

$$\begin{cases} x & x > y \\ y & x < y \end{cases}$$

$$\begin{cases} y & x > y \\ x & x < y \end{cases}$$

$$255 \times sin(\pi \times \frac{x+y}{255})$$

$$255 \times cos(\pi \times \frac{x+y}{255})$$

$$\begin{cases} x & y = 0 \\ x/y\ (mod\ 255) & y \neq 0 \end{cases}$$

$$\frac{x+y}{2}$$

$$0$$

$$1$$

$$255$$

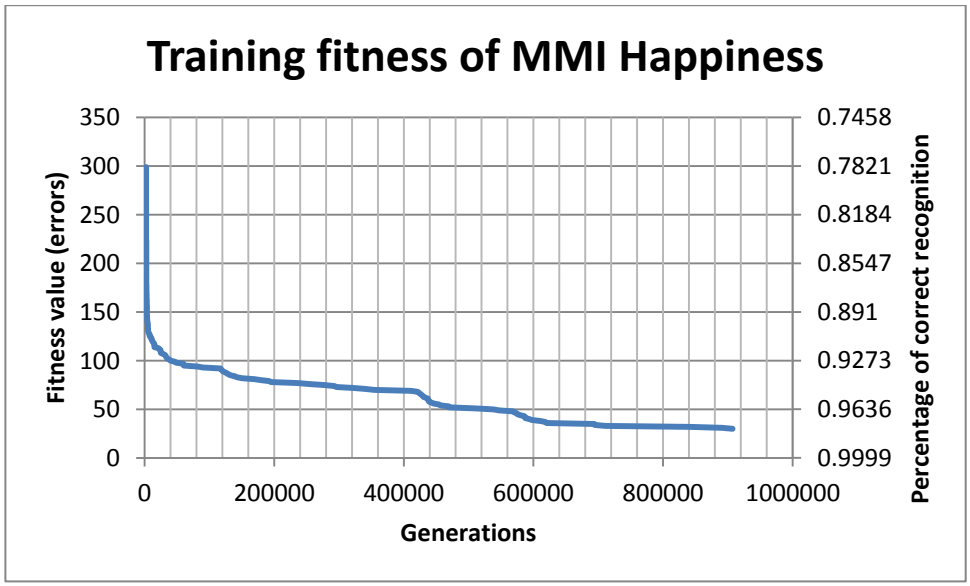
The values are all in float type with range of [0,255). So the input values must be within this range and better evenly distribute. The magnitudes of the flows are usually smaller than 10 pixels with sign. So the flows are truncated by 10 pixels and shift by 10 to make the range to [0, 20]. Then scale to 255 range. The code for this transformation is:

```
if (in > 10)
    in = 10;
else if (in < -10)
    in = -10;
in += 10;
CGPin = in/20.0*255.0;
```

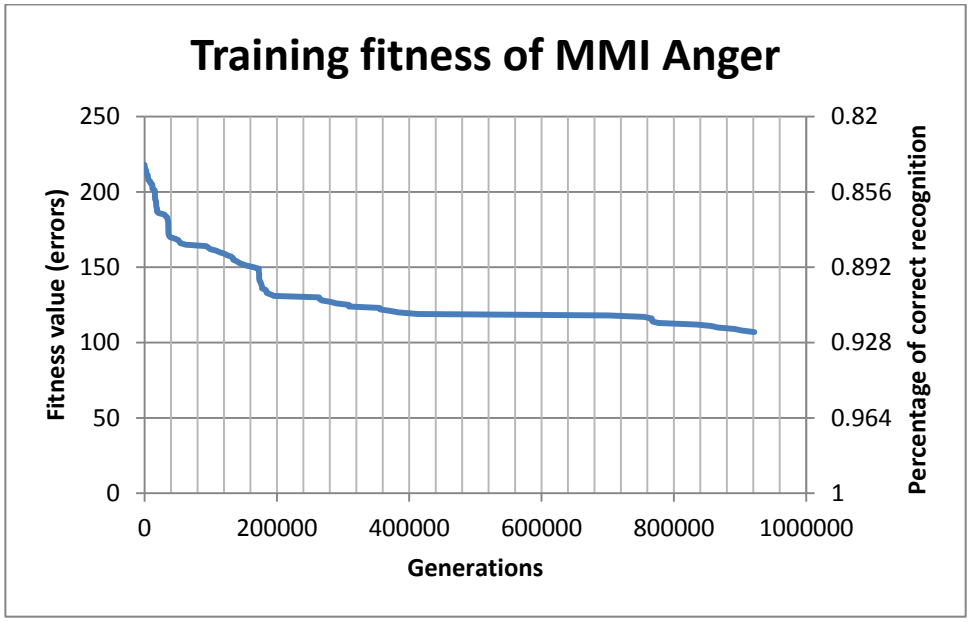
The CGP parameters are same as previous segmentation problem's best setting: 1000 columns, 0.008 mutation rates. Terminal condition: over 1 million generations or fitness value is under 0.1. The fitness value is always integer by this fitness function. Only the value is 0 that can terminate the evolution early than one million generations. And 0 means correctly recognised all input samples, but this is never reached.

4.4.2 The Fitness in Training Process

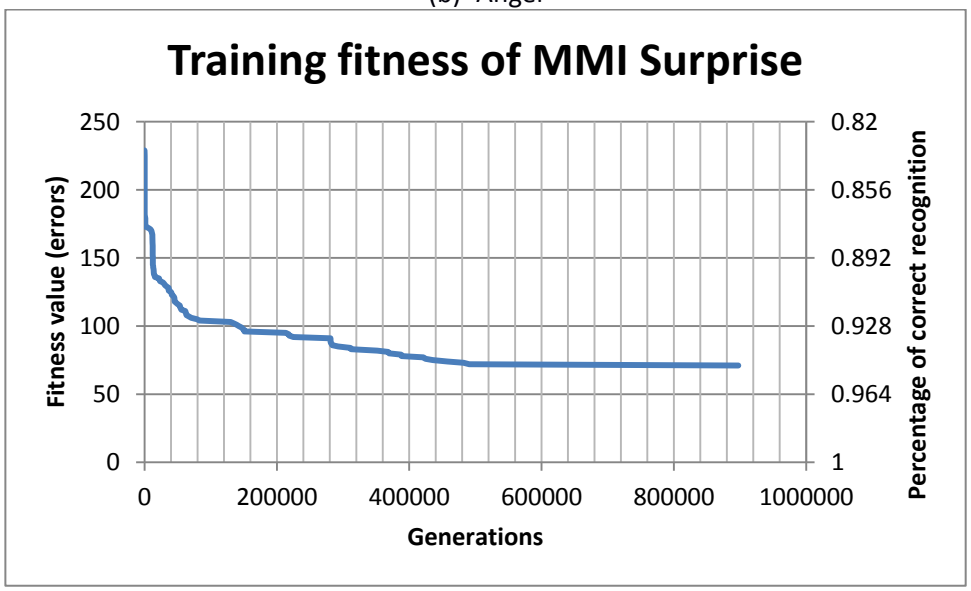
Figure 4.31, 4.32 show the training fitness for each of the expressions in two databases over 1 million generations.



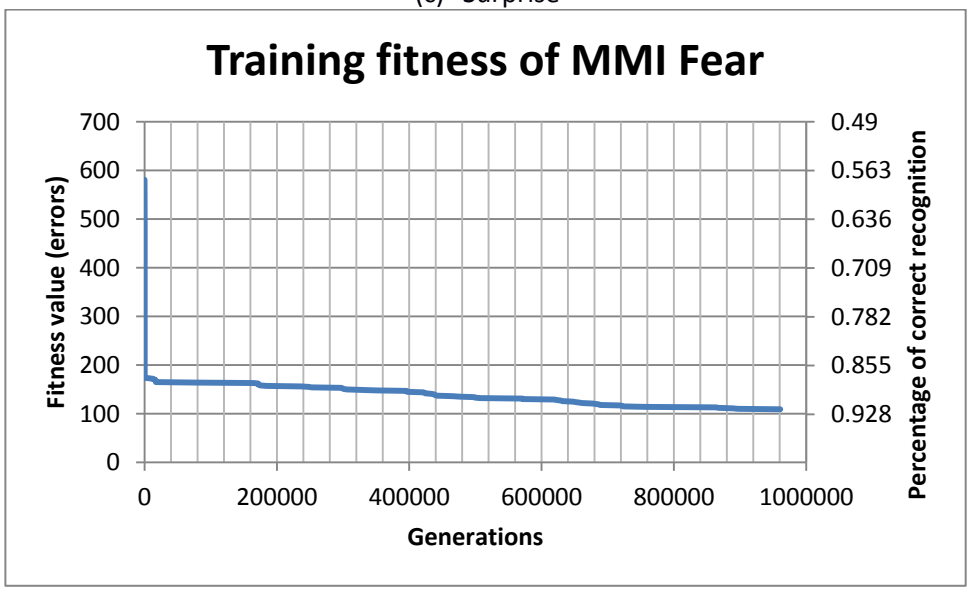
(a) Happiness



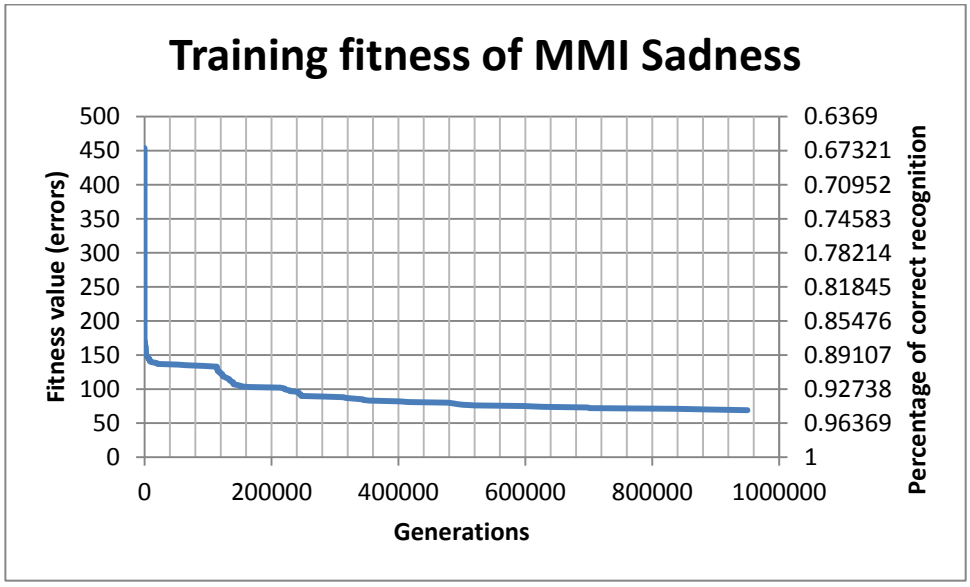
(b) Anger



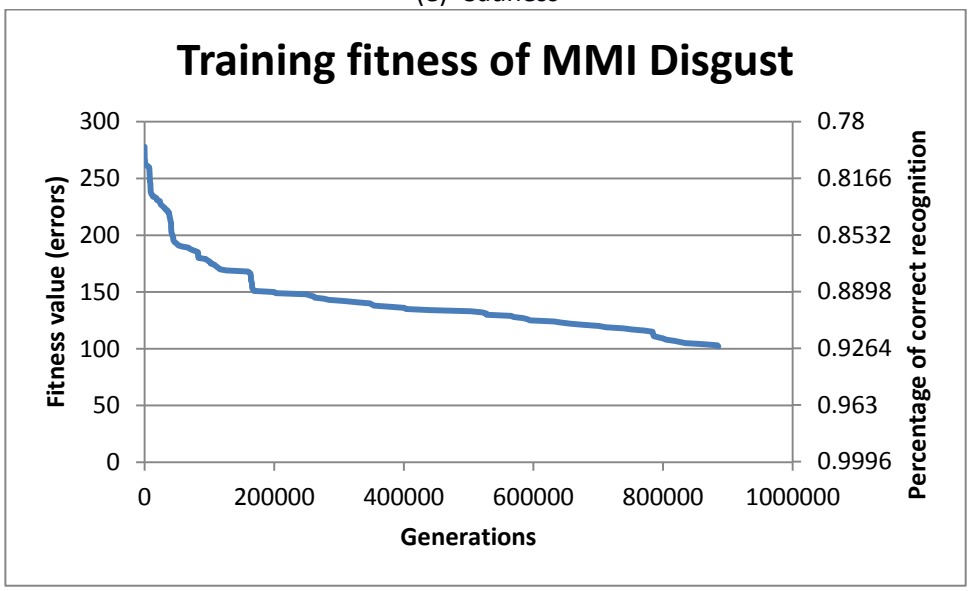
(c) Surprise



(d) Fear

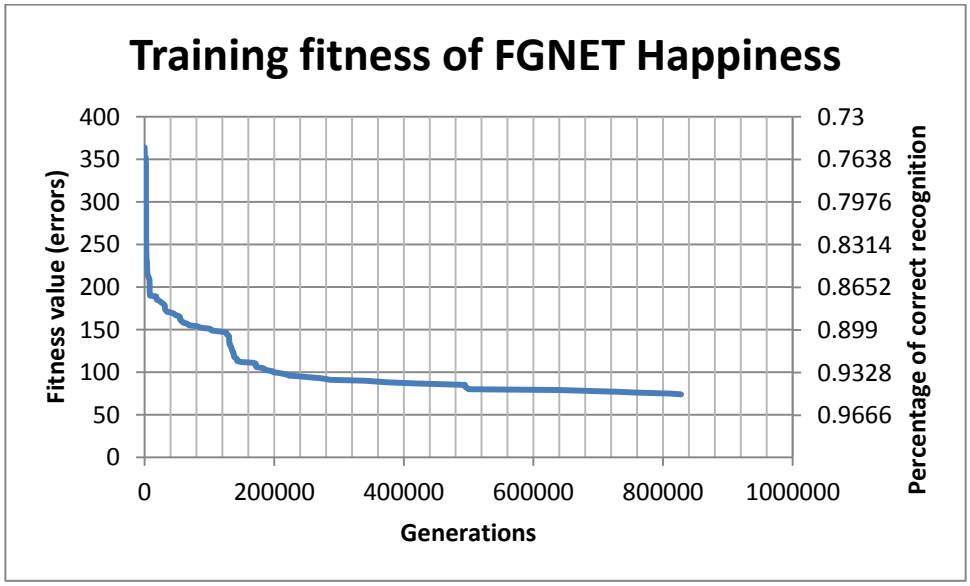


(e) Sadness

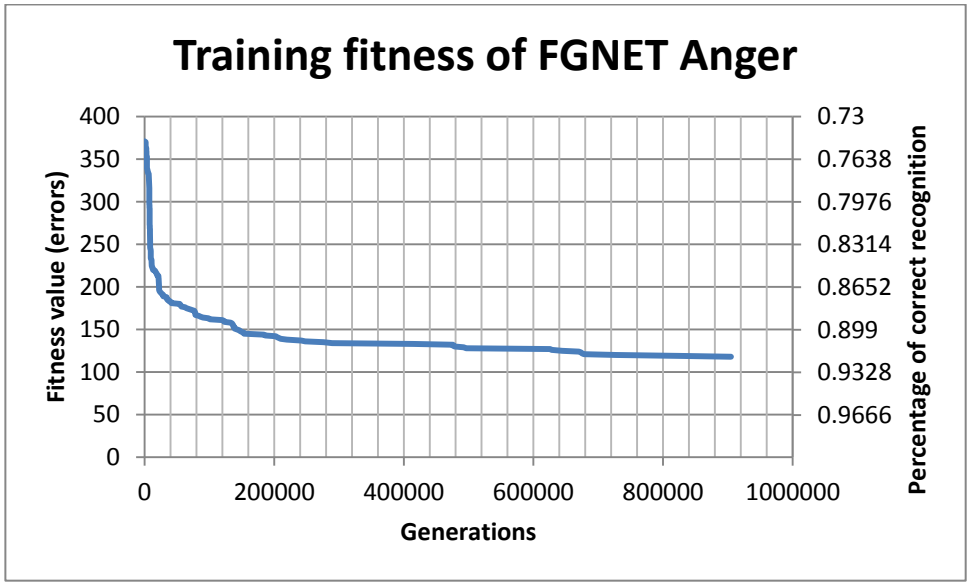


(f) Disgust

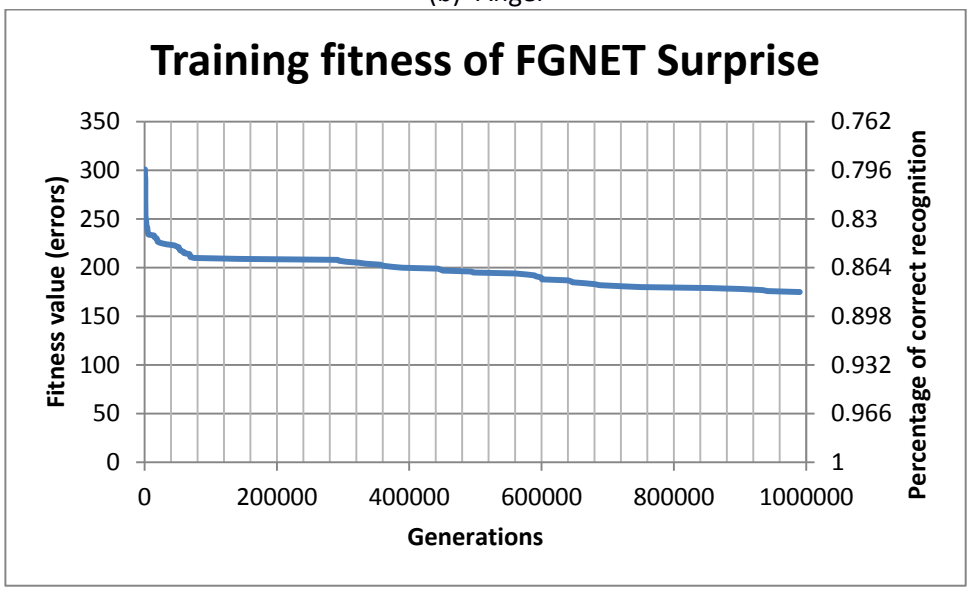
Figure 4.31: The fitness growing in the evolution process (the lower the better), train on MMI database



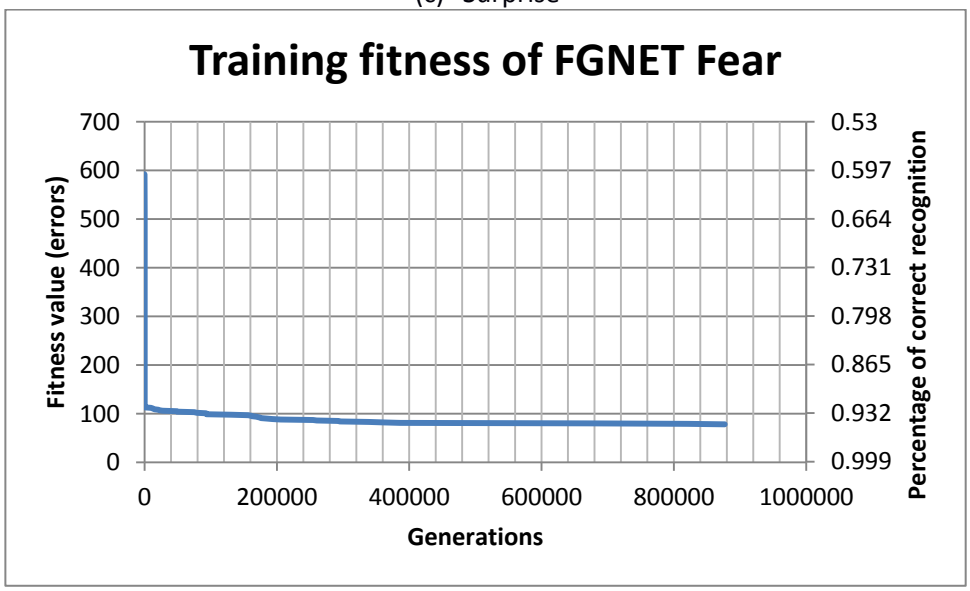
(a) Happiness



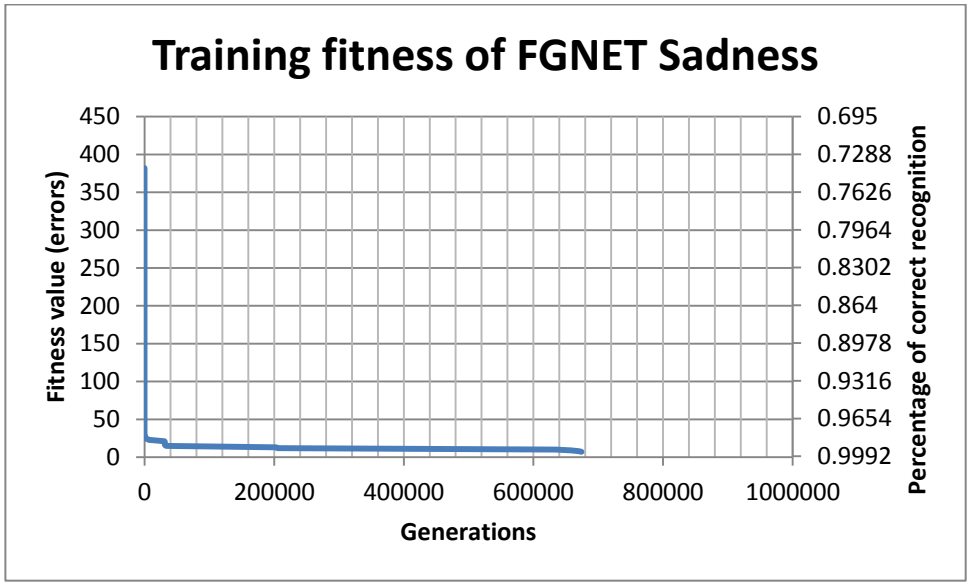
(b) Anger



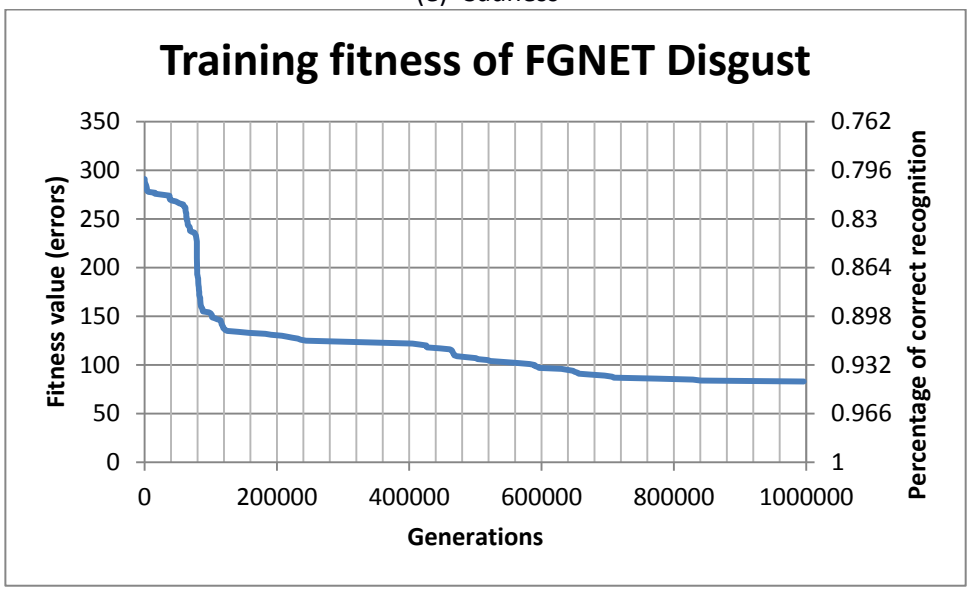
(c) Surprise



(d) Fear



(e) Sadness



(f) Disgust

Figure 4.32: The fitness growing in the evolution process (the lower the better), train on FG-NET database

Follow expected evolution of exponential improvement of fitness, the curves of sadness and fear has significant speed at first few generations. The fact is these two expressions have the most imbalanced dataset and the fitness of the point that evolution goes slow is equal to the number of minority class data. This can be achieved simply full pass all the majority class training data. After that point, the classifier evolves slowly by giving correct minority class classifications.

4.4.3 Experiment Results

Classification results of the CGP classifier, which is trained by MMI database and tested with FG-Net database, are shown in the following table.

Table 4.7: CGP results with fitness function to have most correct recognitions in training (train with MMI, test with FG-NET)

Expression		Happiness	Anger	Surprise	Fear	Sadness	Disgust
Train with MMI	Positive	305	218	230	174	172	278
	Negative	1072	1159	1147	1203	1205	1099
	Fitness	30	107	71	109	69	102
	Percentage	97.82%	92.23%	94.84%	92.08%	94.99%	92.59%
	TP	279	114	162	66	104	178
	TPR	91.48%	52.29%	70.43%	37.93%	60.47%	64.03%
	TN	1068	1156	1143	1202	1204	1097
	TNR	99.63%	99.74%	99.65%	99.92%	99.92%	99.82%
Test by FG-NET	Positive	365	371	303	113	29	292
	Negative	1108	1102	1170	1360	1444	1181
	TP	200	54	58	5	1	44
	TPR	54.79%	14.56%	19.14%	4.42%	3.45%	15.07%
	TN	992	1059	1088	1339	1362	1122
	TNR	89.53%	96.10%	92.99%	98.46%	94.32%	95.00%

The training strategy is: the positive samples are the expression to detect; the negative samples are all other five expressions together. If the numbers of samples are even for every expression, the ratio between two classes in training data would be 1:5 (imbalanced). The movement of natural sad expression is very tiny, so the flows are not obvious. This leads frames for training. The difficulty on cutting transaction of sadness makes the situation worse, the shortage of sadness samples makes the ratio 1:30 in FG-NET. The testing results show the more the data is imbalanced the lower true positive rates are. The happiness is easier to classify than other expressions, the TPR achieves 91% on training and 55% on testing. Other expressions also have about 50% rate drop of TPR from training to testing. In the experiments, the higher training fitness is not related to better performance.

Table 4.8: CGP results with fitness function to have most correct recognitions in training (train with FG-NET, test with MMI)

Expression		Happiness	Anger	Surprise	Fear	Sadness	Disgust
Train with FG-NET	Positive	365	371	303	113	29	292
	Negative	1108	1102	1170	1360	1444	1181
	Fitness	74	118	175	78	7	83
	Percentage	94.63%	91.43%	87.29%	94.34%	99.49%	93.97%
	TP	306	292	132	35	22	209
	TPR	83.84%	78.71%	43.56%	30.97%	75.86%	71.58%
	TN	1093	1063	1166	1360	1444	1181
	TNR	98.65%	96.46%	99.66%	100.00%	100.00%	100.00%
Test by MMI	Positive	305	218	230	174	172	278
	Negative	1072	1159	1147	1203	1205	1099
	TP	190	112	19	7	8	76
	TPR	62.30%	51.38%	8.26%	4.02%	4.65%	27.34%
	TN	955	908	1070	1188	1167	991
	TNR	89.09%	78.34%	93.29%	98.75%	96.85%	90.17%

The similar thing happens when training with FG-NET database and testing with MMI database. The training fitness is around 90%, four of TPRs of training data are above 70%, but the testing TPR drops badly. Two of them, which are the most imbalanced, have the true positive rates under 5%.

In the imbalanced training process, the fitness just relies on the correct classification number. The correct numbers on major class and minor class have the same weight. The classifier does not consider the accuracy of minor class. Assume the case if there is one positive sample in the training data set, then the training classification result will be easily 99% by returning full negative signals. The TNR is 100%, but the TPR is 0%. So for the imbalanced training problem, the accuracy should be considered during the training phase. Otherwise the classifier is not likely to have good performance.

The fitness can achieve a quite high value. When the experiments in previous section reach high fitness, the classification result should achieve a corresponding high rate. However the results in table 4.7, 4.8 show that the performances are rather poor, except the happiness, even on the training data. Because the ultra-imbalanced dataset makes that the accuracy is very low on minor class. Classification on imbalanced data is an existing problem to classifiers. It is common that they all perform worse on imbalanced data than balanced ones. There are several ways to deal with this problem. One way possible way is working on the number of data, to make the data balance: down sample the data of majority class, resample the data of minority class. But for CGP, the fitness function can be adjusted to solve this problem.

4.5 Fitness Function of AUC

The fitness function can be studied to solve the imbalanced data classification problem. It should also be simple and straight forward, easy to get better a small bit every time, this means the direction should be simple, clear and specific, instead of tough or abstract conditions. Then the area under curve (AUC) is considered (because of its easy to evolve/improve and not affect by the number of data).

4.5.1 Receiver Operating Characteristic

The receiver operating characteristic is the graph to describe the performance of a binary classification. The point on the graph means a state under a certain threshold of the classifier. The location of one point is decided by the true positive rate and the false positive rate.

Let us consider a two-class prediction problem (binary classification). The outcomes are labelled either as positive (p) or negative (n). There are four possible outcomes from a binary classifier. If the outcome from a prediction is p and the actual value is also p, then it is called a true positive (TP); however if the actual value is n then it is said to be a false positive (FP). Conversely, a true negative (TN) has occurred when both the prediction outcome and the actual value are n, and false negative (FN) is when the prediction outcome is n while the actual value is p.

Table 4.9: The possible outcomes of binary classification

		Real		SUM
		p	n	
Predicted	p'	TP	FP	P'
	n'	FN	TN	N'
SUM		P	N	

True positive rate (TPR) = TP/P = TP/ (TP+FN) (4.6)

False positive rate (FPR) = FP/N = FP/ (FP+TN) (4.7)

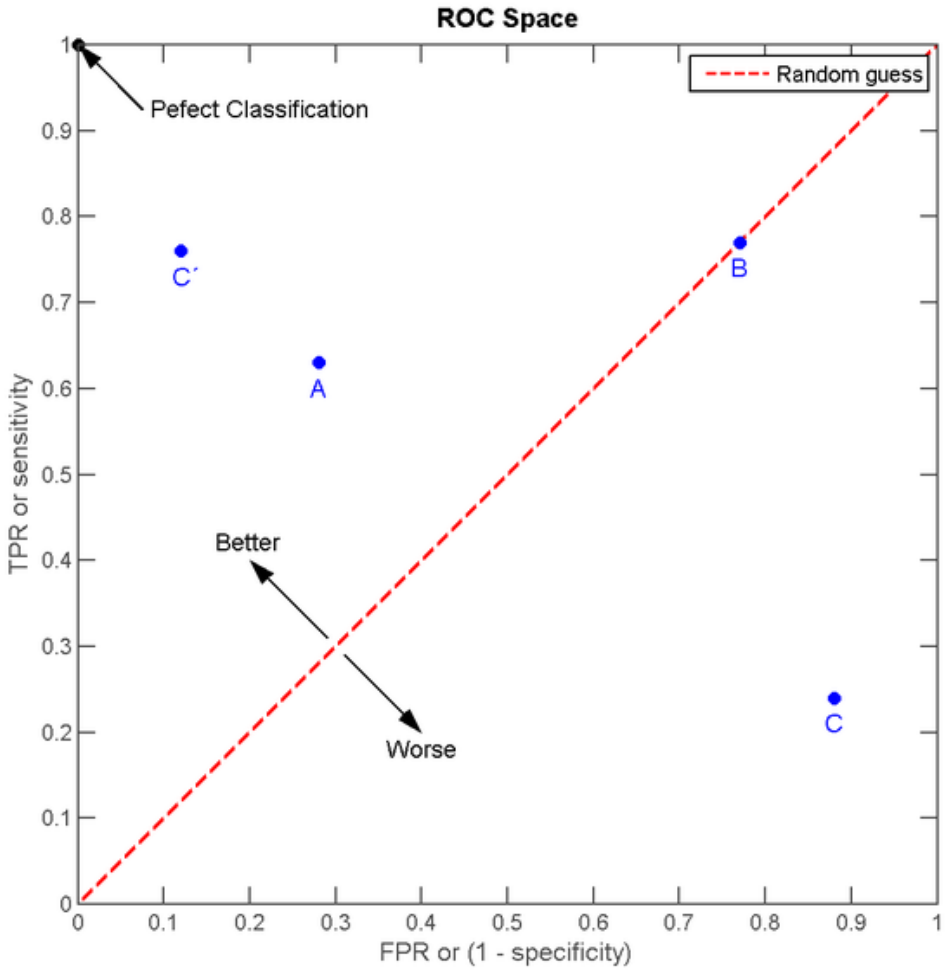


Figure 4.33: The ROC curve [140]

The (0,0) point is non-pass classifier, always return negative label. And (1,1) point means all pass classifier, so all data will be recognised as positive. The (0,1) point on the top left corner means perfect classification. The diagonal line means the classifier give the result by a randomly 50% chance. The points above the diagonal line indicate the classifier is better than guessing. The points under the diagonal line can simple switch the classification results to make it better than random. The ROC curve is obtained by shifting the threshold from non-pass to all-pass, the points of TPR and FPR of each thresholding form the curve. It is noticeable that the TPR and FPR are not related to the training sample numbers or the balance of samples of two classes.

4.5.2 Area under Curve

Area Under the Curve[141]: as the points higher than the diagonal line are better, the bigger area under the curve the better classifier performance is. So the AUC is possible to evaluate the performance of the classifier based on two major reasons:

a) AUC is not affected by the imbalanced data.

b) The area is easy to improve a small step every generation.

4.5.2.1 Trapezoidal Rule for AUC Calculation

The calculation of AUC is actually the important thing. The area is within a square with length of 1 stands for the rate between 0% and 100%. The area of a perfect classification is always 1. The random guess has the area of 0.5 (diagonal line, half of the whole area). The AUC of a classifier should be within 0.5 and 1. The real ROC curve is continuous but finite points are used to estimate the AUC, the number of points is no more than number of sample data.

Trapezoidal rule is often used for the estimation as shown in the following figure.

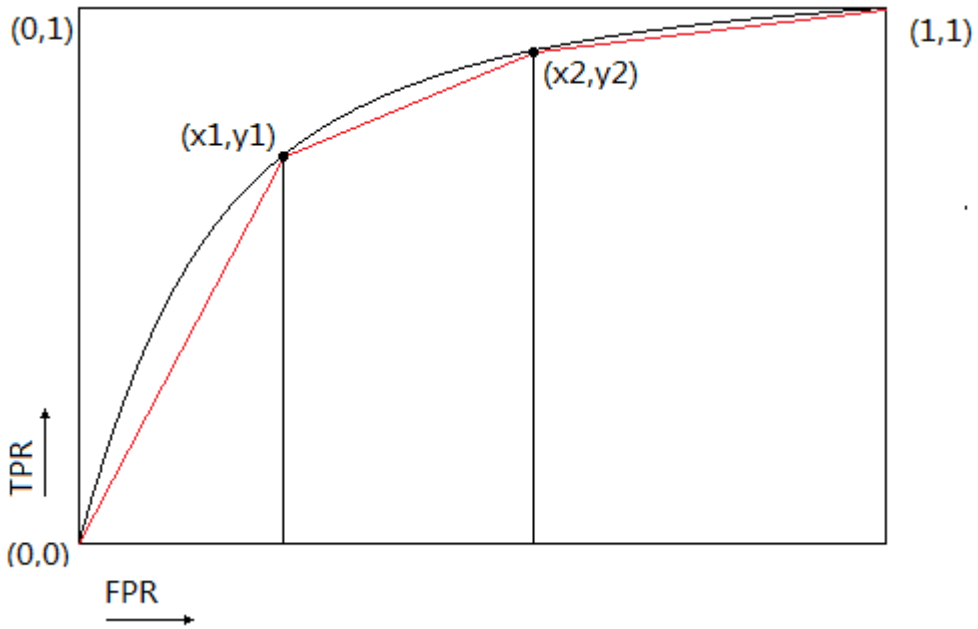


Figure 4.34: Trapezoidal rule for estimating AUC

The black curve is the real ROC, we have two points (x_1,y_1) and (x_2,y_2) where x means false positive(FPR) rate and y stands for true positive rate(TPR). One point on the curve means the

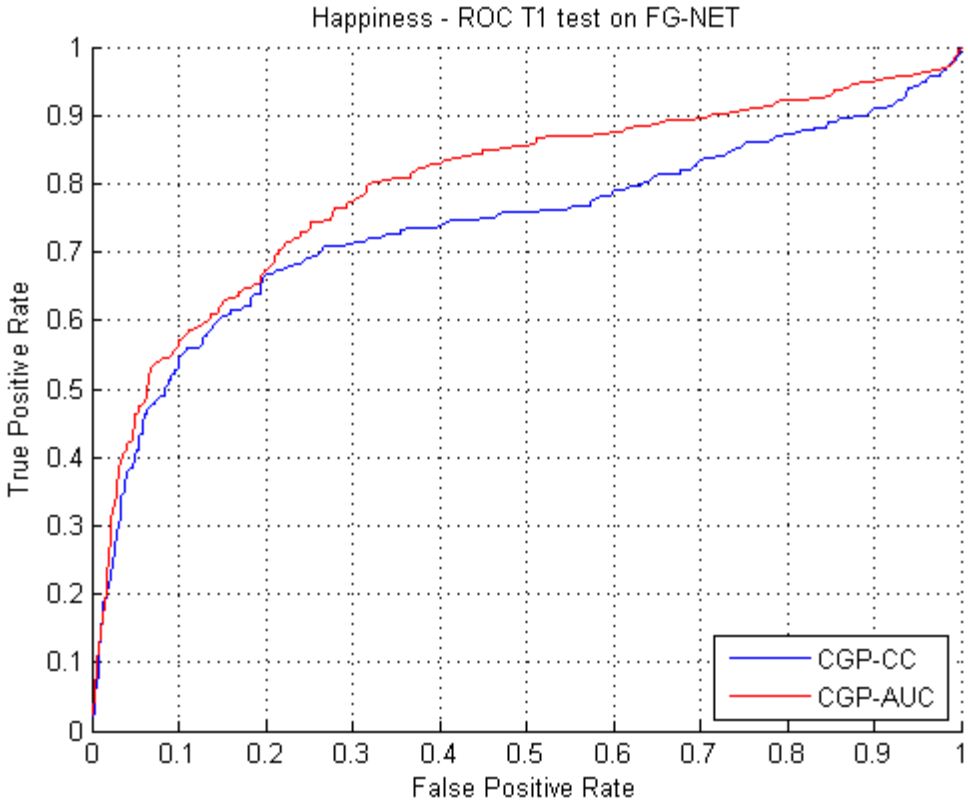
performance at the unique classifier threshold. If connect the adjacent point with straight lines (red lines in figure), then the AUC is estimated from the sum of trapezoidal areas.

$$AUC \approx \sum_{i=0,n-1}(x_{i+1} - x_i)(y_{i+1} + y_i)/2 \qquad (4.8)$$

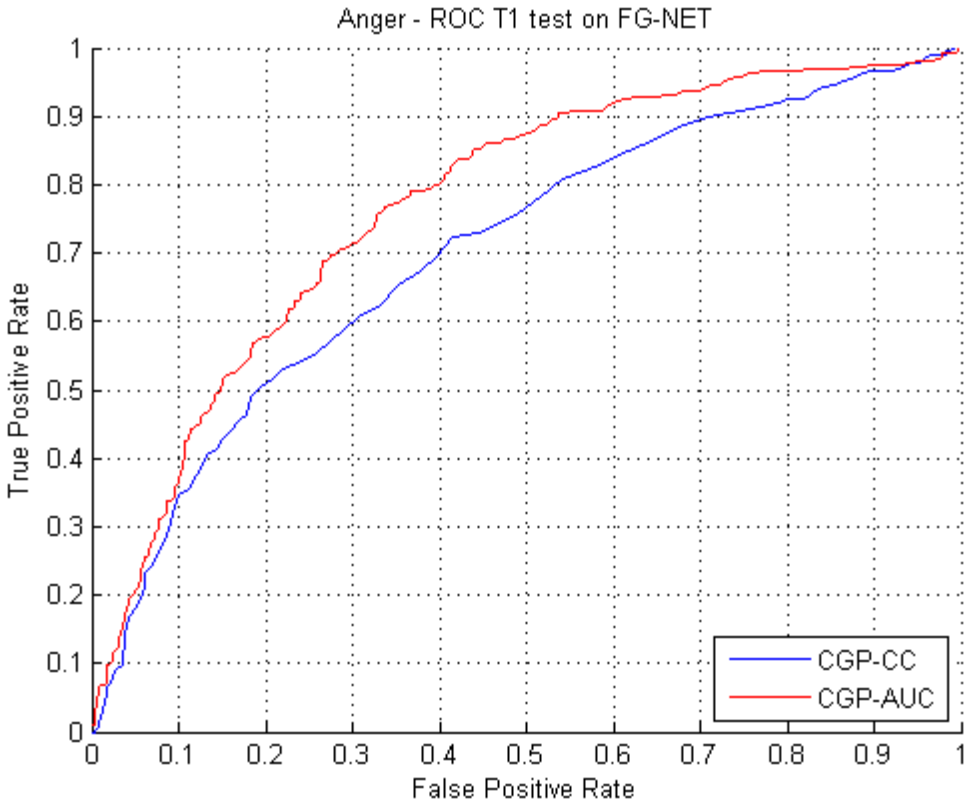
4.5.3 Results by Using AUC Fitness Function

Parameters: 1000 columns, 0.008 mutation rates, 1 million generations, fitness 0.01 (never reached). Here is the ROC curve of CGP classifier results on testing FG-NET data after trained by MMI database. The video clips are cropped manually that only the face area remains. The rising parts of expressions are cut and dense optical flows are extracted between one frame and the fifth after. Each whole picture's dense optical flow is one sample and the total numbers of samples are in table 3.1 and 3.2. Then the flow data are reduced by averaging the grids, the image is divided into 24 equal grids (T1). There are six binary classification experiments for six expressions. Each experiment requires the positive samples, the samples of corresponding expression, and the negative samples, samples of all other five expressions. The numbers of samples for experiments are shown in table 3.8.

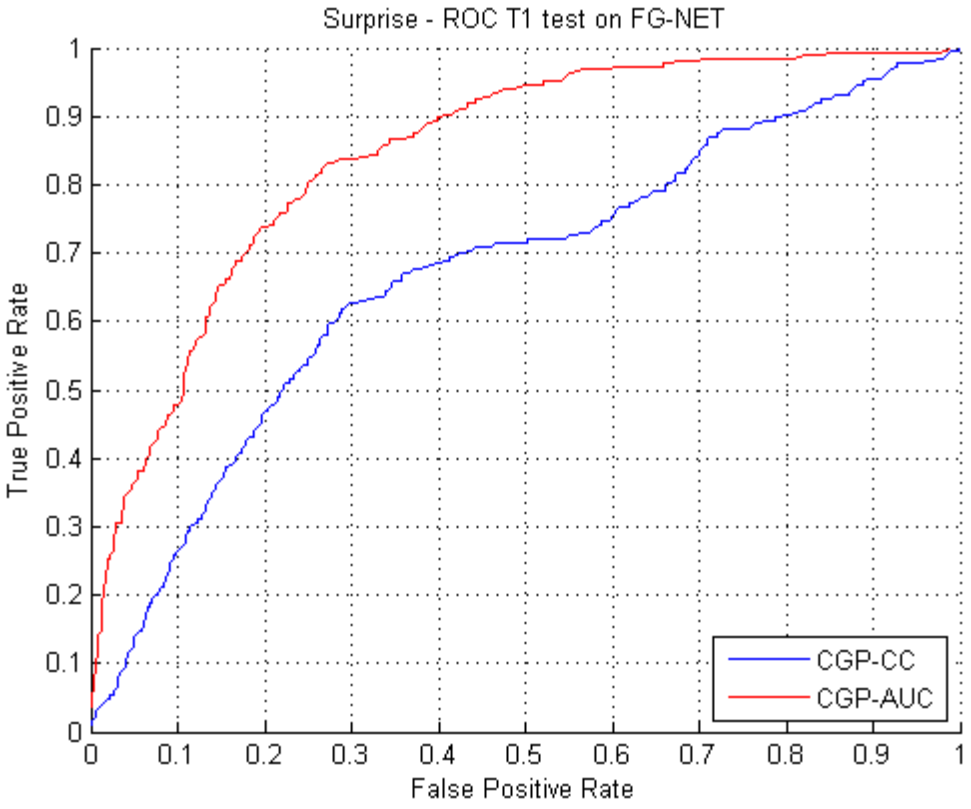
The results of CGP classifier with AUC based fitness function (CGP-AUC) are shown by the ROC curves and compared with previous experiments curves, which are trained with the fitness function of counting the correct classifications (CGP-CC).



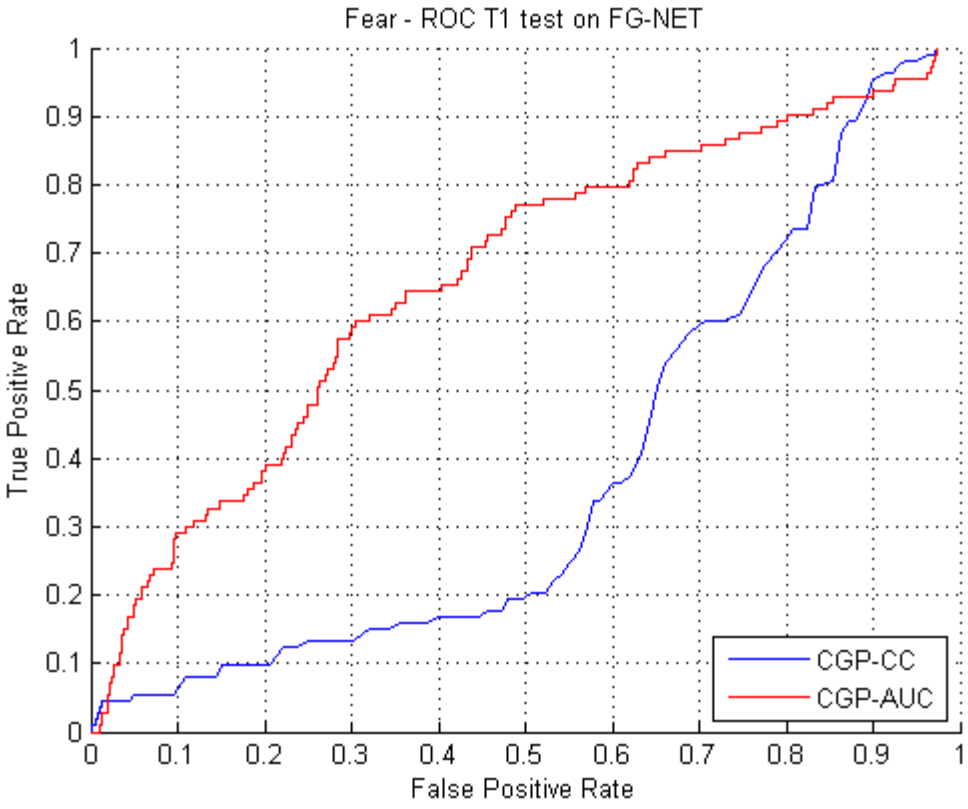
(a) Happiness



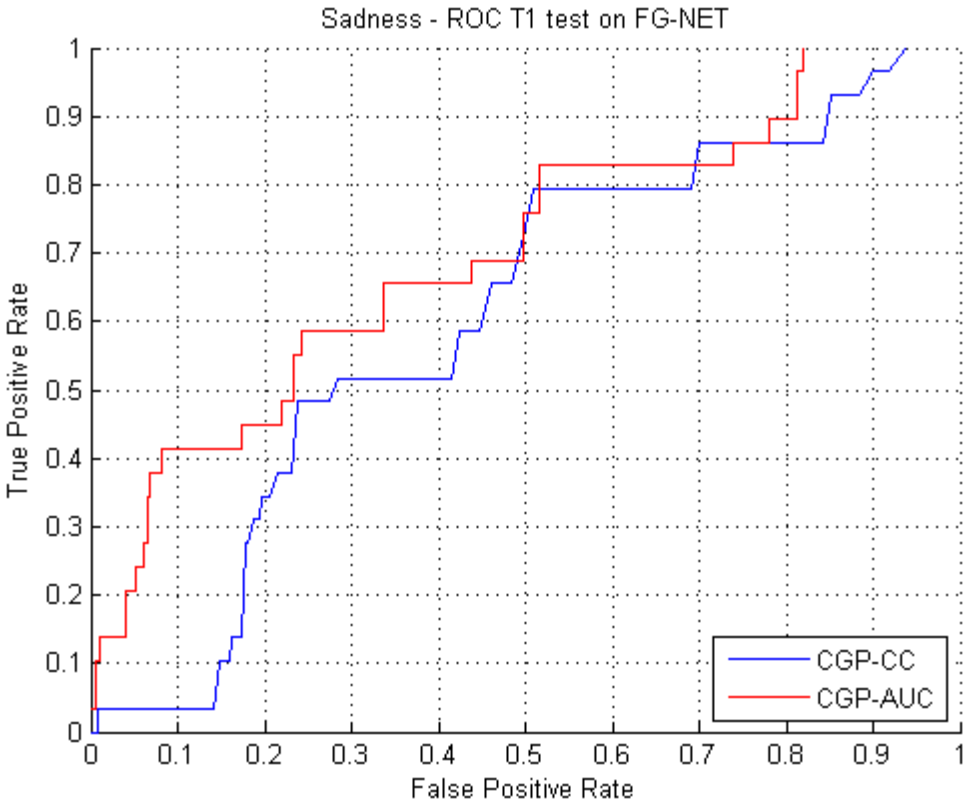
(b) Anger



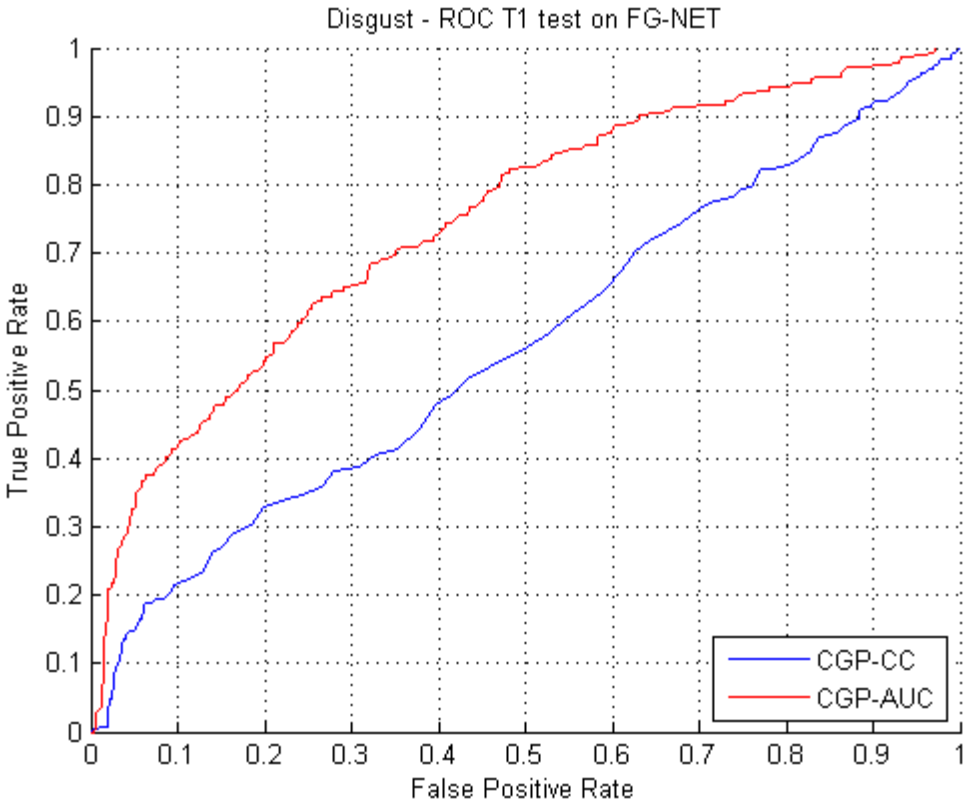
(c) Surprise



(d) Fear



(e) Sadness



(f) Disgust

Figure 4.35: Comparison of two fitness functions' results

Where the blue curves are ROC of the CGP classifier trained by AUC fitness, the red curves are from the performances of classifiers with counting correct classification fitness function.

Each point on the curve is a classification performance with the location of TPR and FPR. The AUC fitness function helps the classifier performs much better than the other fitness function in this imbalanced dataset. The results show the counting correct classifications fitness function may have worse performance than random guess on this problem.

The actual threshold is selected by user depends on the situation. The curve of sadness is not smooth, that is because the samples for testing are very few. All the AUCs of training samples are above 0.98, but the testing results of another database are not good enough. The reasons include:

a. The differences between two database

b. The way of sampling data used for training and testing (this will be discussed at the end of this chapter)

c. And the possibly overfitting caused by the long evolution process.

4.6 Overfitting

4.6.1 Introduction

In the training process, a classifier learns from the samples with known labels. But in the realistic problem, samples are not possible to be perfect, but will contain errors or noises. If the classifier fits the training data too well, then it will not perform well on unseen test data.

To reduce the overfitting on CGP classifier, the evolution process should be finished before the classifier goes to overfit. There are two ways to shorten the training process:

a. Reduce the generation
b. Reduce the restriction of terminal fitness

They are both loose the terminal conditions to let the evolution finish earlier. A series of experiments on the terminal conditions have been made including generations: 3,000, 6,000, 10,000, fitness: 0.05, 0.1, 0.15, 0.2. Then the data is massive that is not convenient to show all of them here. Instead, the 10K generations and 0.1 fitness terminal conditions are used here as the comparison to 1 million generation evolution. They have run 10 times and the AUC values on the testing data, which is the different database from the training data, are used for the comparison. They include 24 grids, 48 grids, and 96 grids (shown as T1, T2 and T3). The results will be presented in box plots; therefore the next section explains the box plot in first.

4.6.2 Box Plot

Box plot[142] is commonly used in statistic for the analysis of data distribution. There are four kinds of points on a box plot:

a. Bottom of box(first quartile, split the lower 25% of data)
b. Median(second quartile, split the 50% of data)
c. Top of box(third quartile, split the higher 25% of data)
d. Maximum or minimum value within 1.5 IQR
e. Points outside 1.5 IQR (denoted as a circle)

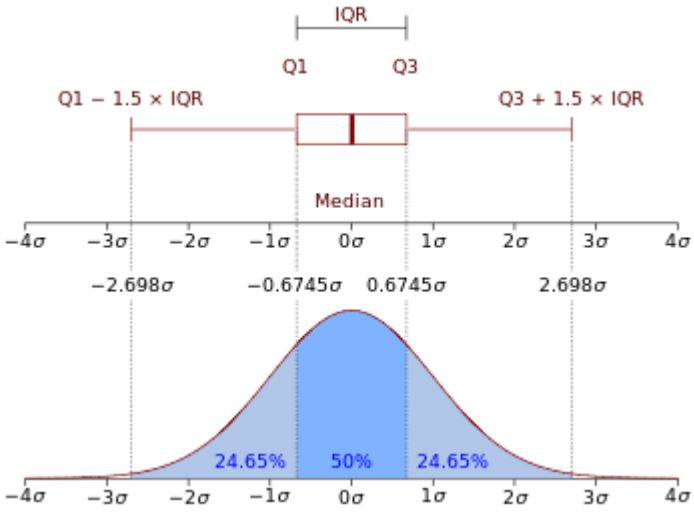


Figure 4.36: Explanation of box plot [143]

The percentage for quartile splitting is not the value of data but the number of data. IQR is interquartile range, which is the value of distance between first and third quartile.

Box plot of 24 Grids (T1)

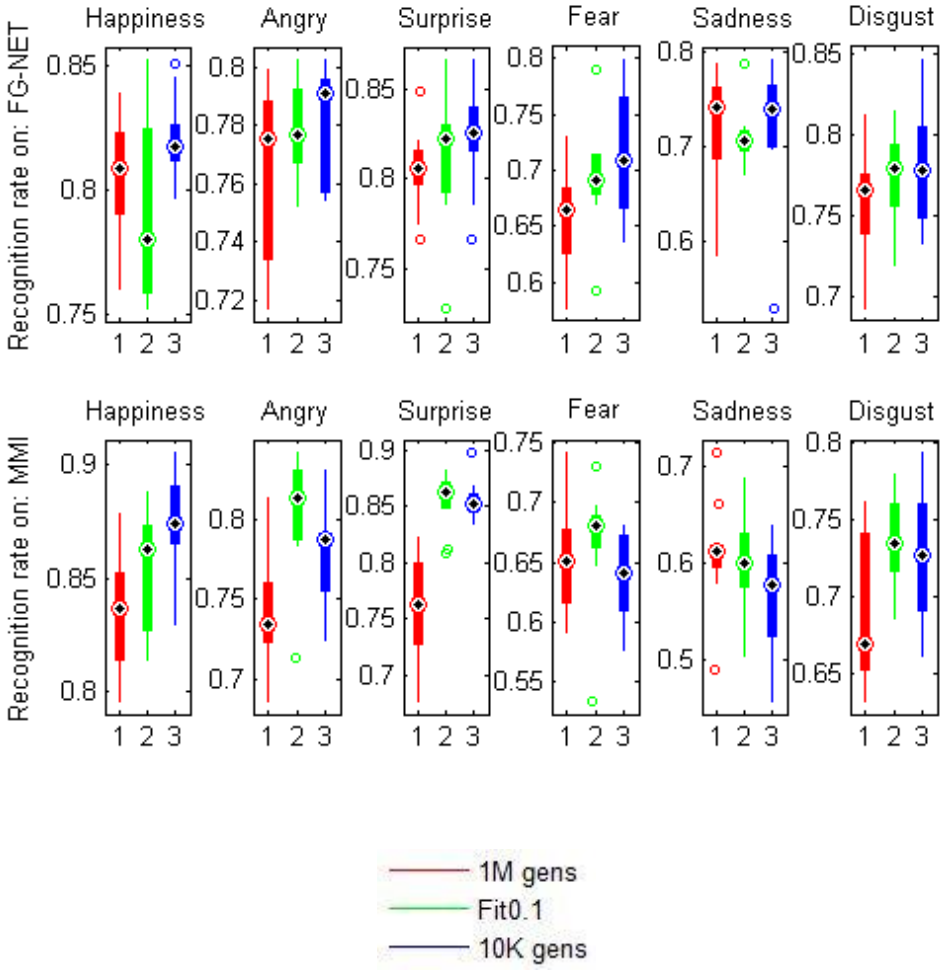


Figure 4.37: Box plot of accuracies when test on the other database in ten runs (T1)

Box plot of 48 Grids (T2)

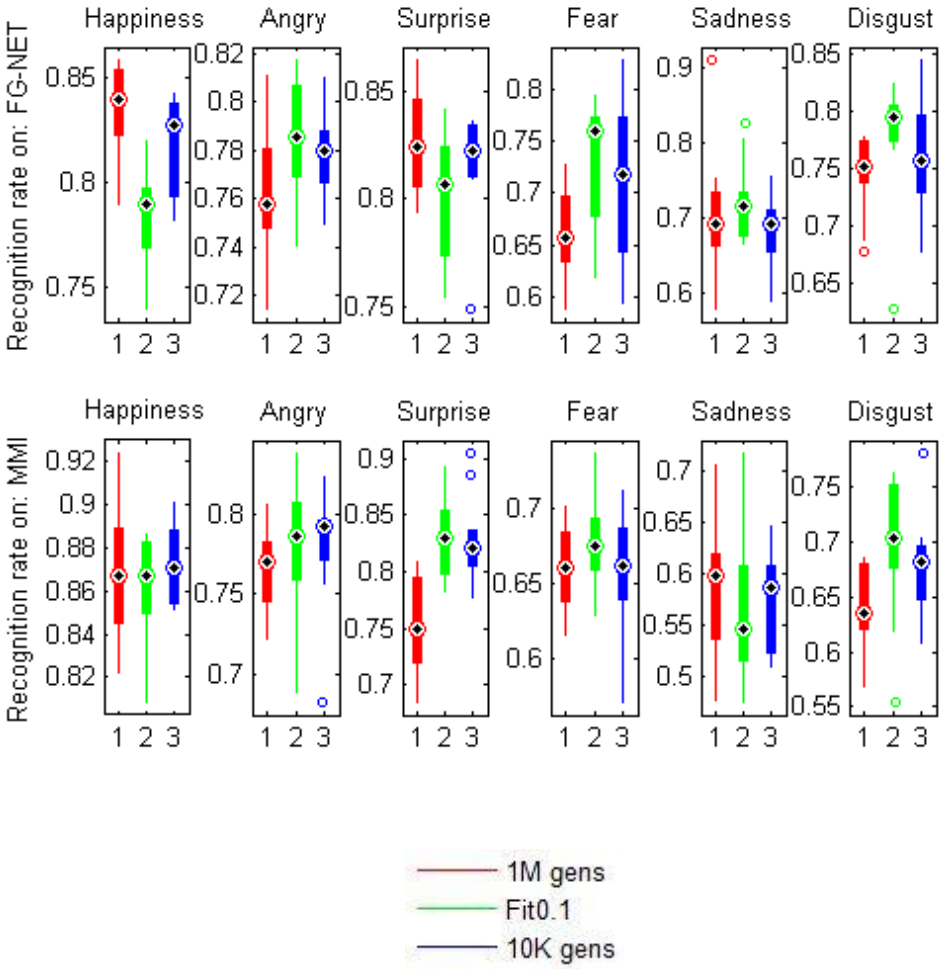


Figure 4.38: Box plot of accuracies when test on the other database in ten runs (T2)

Box plot of 96 Grids (T3)

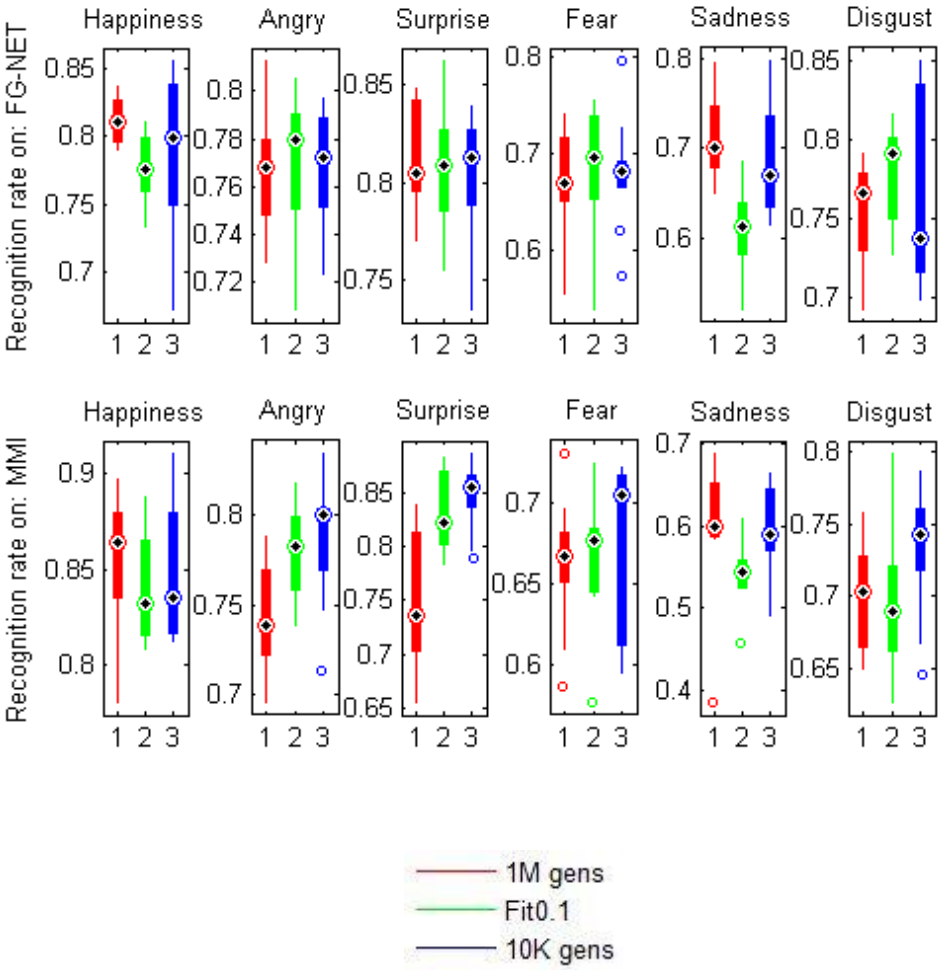


Figure 4.39: Box plot of accuracies when test on the other database in ten runs (T3)

Figures 4.37-4.39 are the box plot of the accuracies of CGP classifier with different terminal conditions in ten runs. One figure is a series of experiments with one grid setting, from 24 (T1), 48 (T2) to 96 (T3). Each figure has two rows, the upper row is the box plot of the results when train by MMI database and test with FG-NET database. The lower row is the opposite, train by FG-NET and test by MMI. The three colours mean three different terminal conditions: red is 1 million generations, green is fitness reach 0.1 and blue is 10,000 generations maximum evolution.

From figures 4.37-4.39, the blue boxes are higher than red ones approximately 66% of the time. So the shorter generation is better than the very long generation (one million generations). Not only the 10K generations, but 3K and 6K generations' results are also not weaker than 1 million generations' result. The running time also has 100 to 300 times differences. The 0.1 fitness terminal condition has 1/3 time higher boxes than one million generation results, but also 1/3 time worse than them. When test on MMI database T3 grids and test on FG-NET T1, the 10K

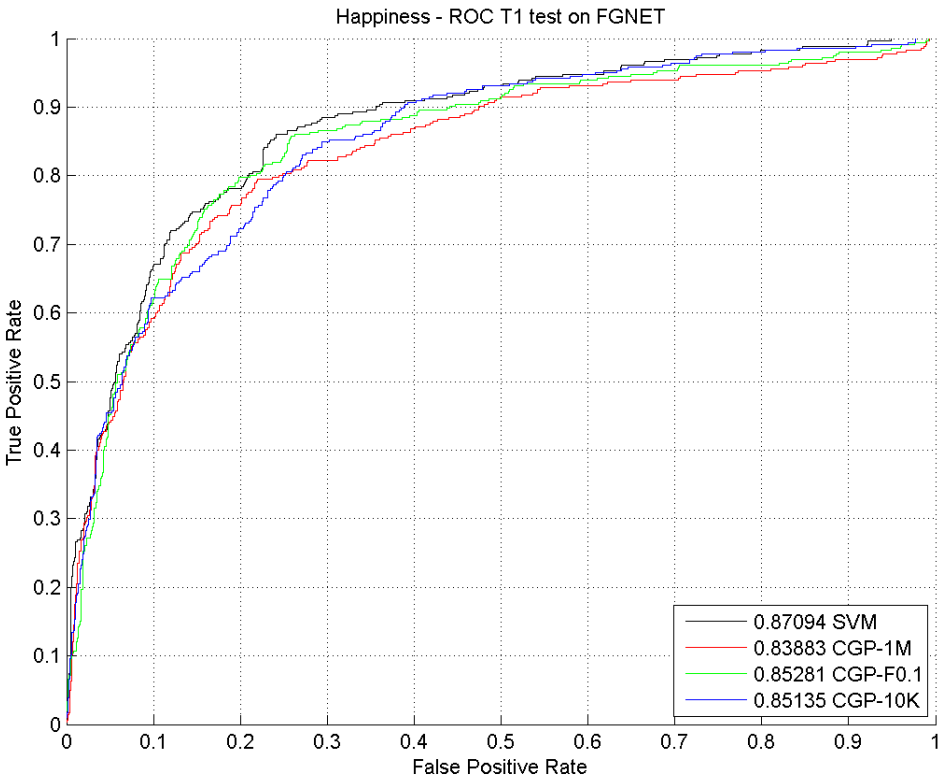
generations result is obviously better than 0.1 fitness terminal condition. In other cases, one could not win the other, so overall blue boxes are slightly better than green boxes. In other experiments, which results are not listed here, 0.01 fitness results are not too much different from 1 million generations' results and they usually finish at around 2K to 5K generations. 0.2 fitness results are worst among all the experiments, they usually finish under 1.5K generations. So shorten the generations for evolution is efficient and can achieve good results. Setting a larger terminal condition of fitness value is not stable. Sometimes it has superior performance, but sometimes result is too bad.

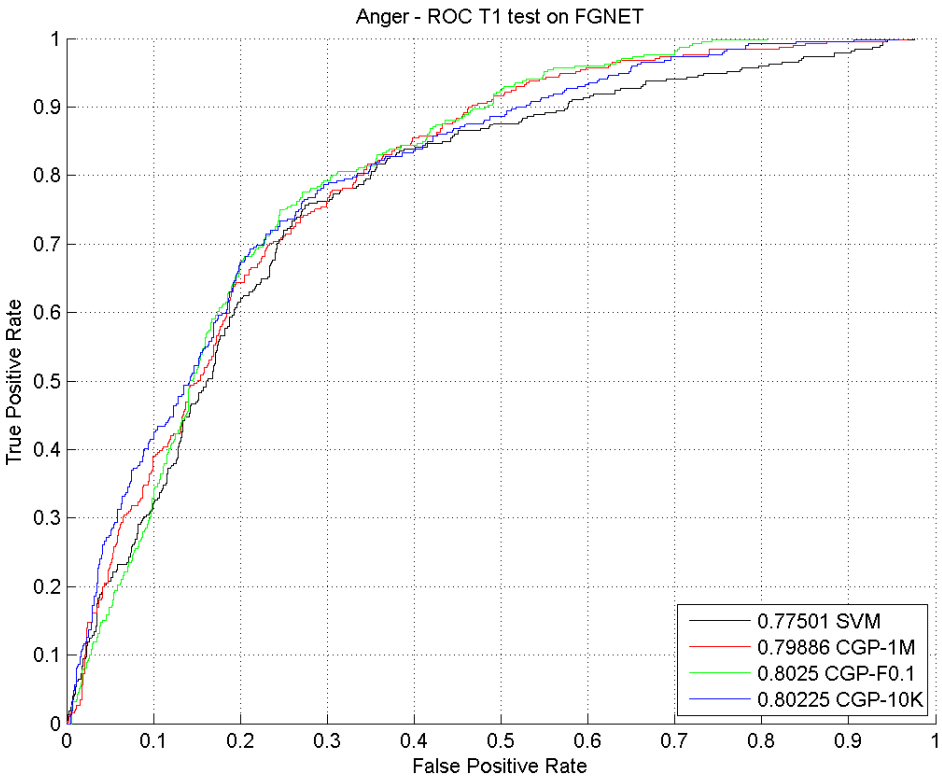
4.6.3 Comparison between CGP and SVM

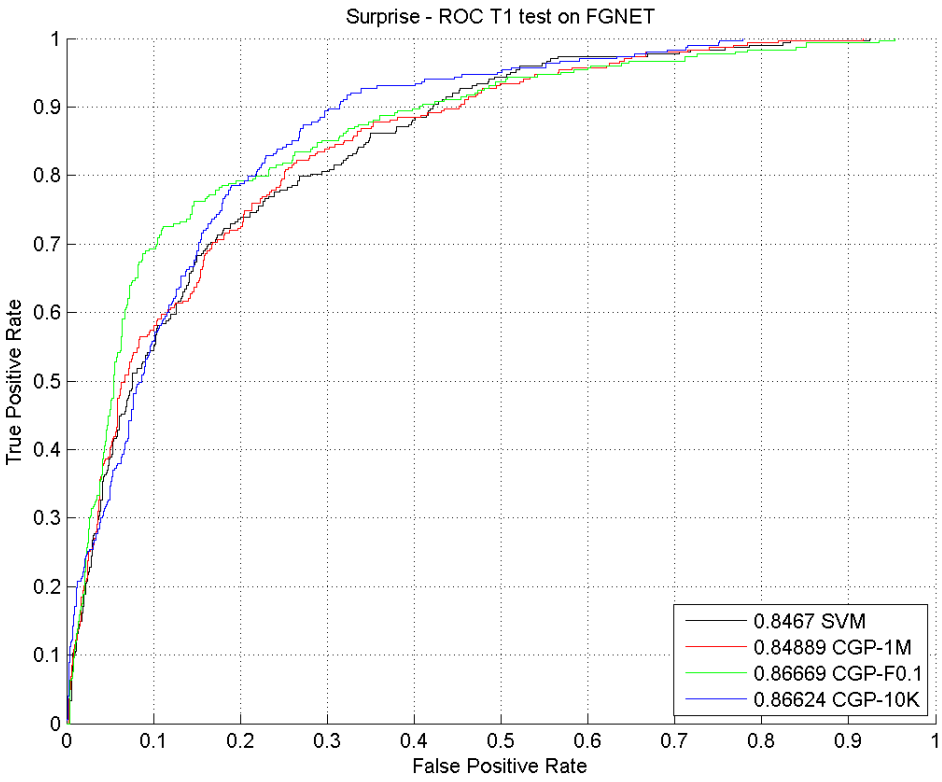
Then the comparisons between CGP and SVM are going to make in this section. The comparison will be on the value of AUC and ROC from previous experiments' results. The results of CGP classifier are from section 4.6.2 and the SVM results are from section 3.5.3.2 a). Three CGP classifiers are selected for the comparison, they are CGP with 1 million generations (CGP-1M), CGP with terminal condition of fitness less than 0.1 (CGP-F01) and CGP with 10 thousand generations (CGP-10K). The curves of different classifiers will be printed at their own colours: SVM is black; CGP-1M is red; CGP-F01 is green; CGP-10K is blue.

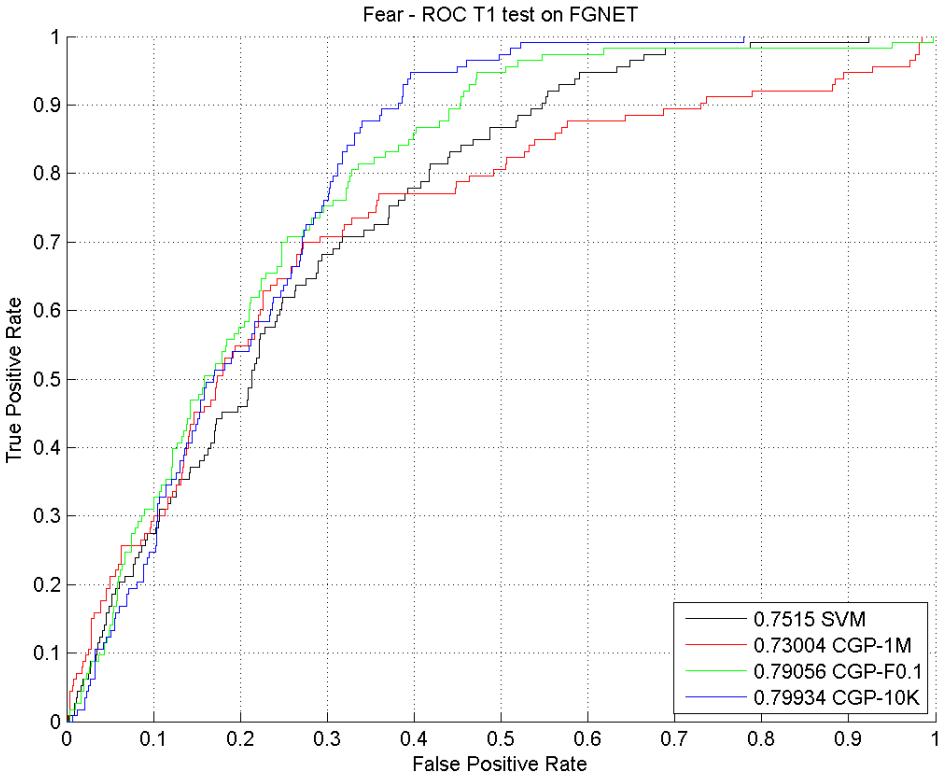
4.6.3.1 AUC Graph of 24 Grids (T1)

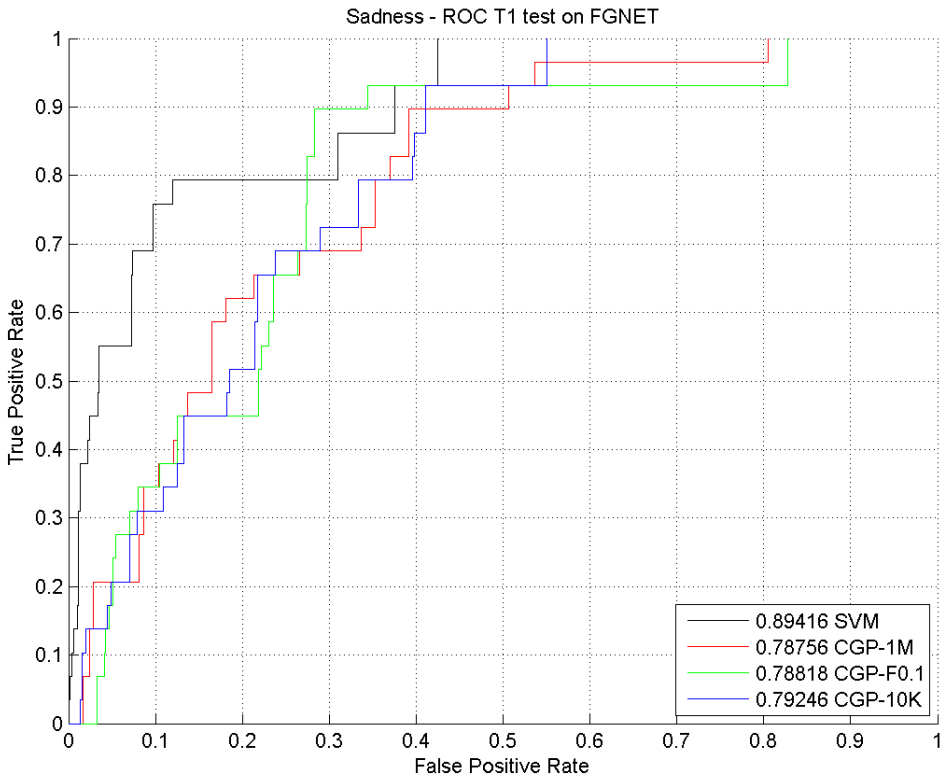
a) Trained by MMI, test with FGNET






Surprise - ROC T1 test on FGNET
True Positive Rate
False Positive Rate
0.8467 SVM
0.84889 CGP-1M
0.86669 CGP-F0.1
0.86624 CGP-10K


Fear - ROC T1 test on FGNET
True Positive Rate
False Positive Rate
0.7515 SVM
0.73004 CGP-1M
0.79056 CGP-F0.1
0.79934 CGP-10K



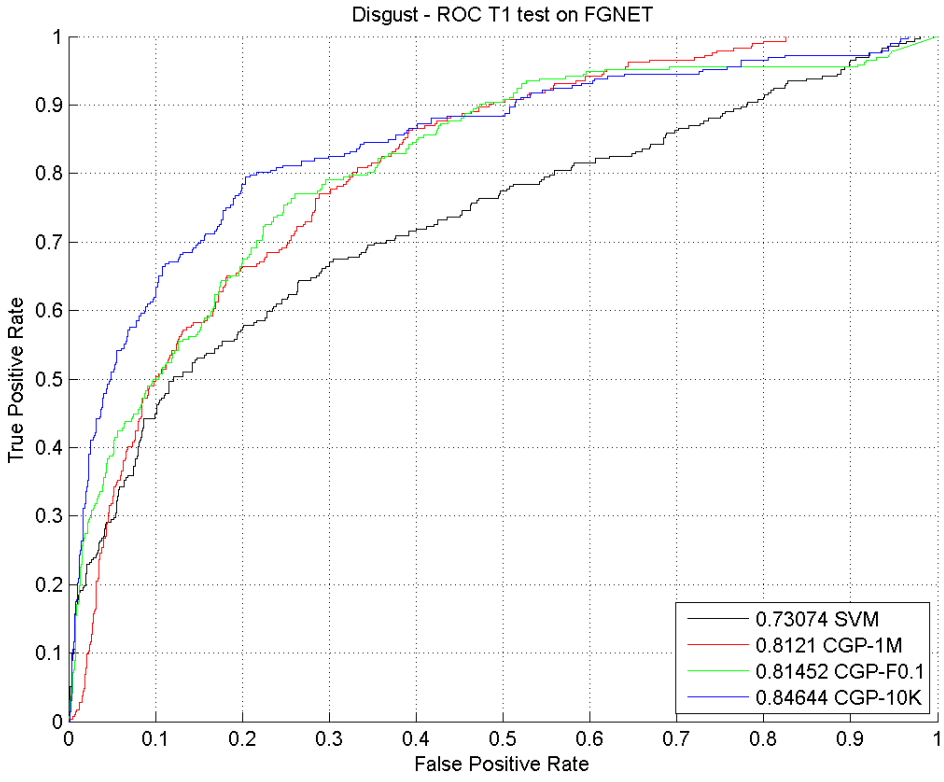
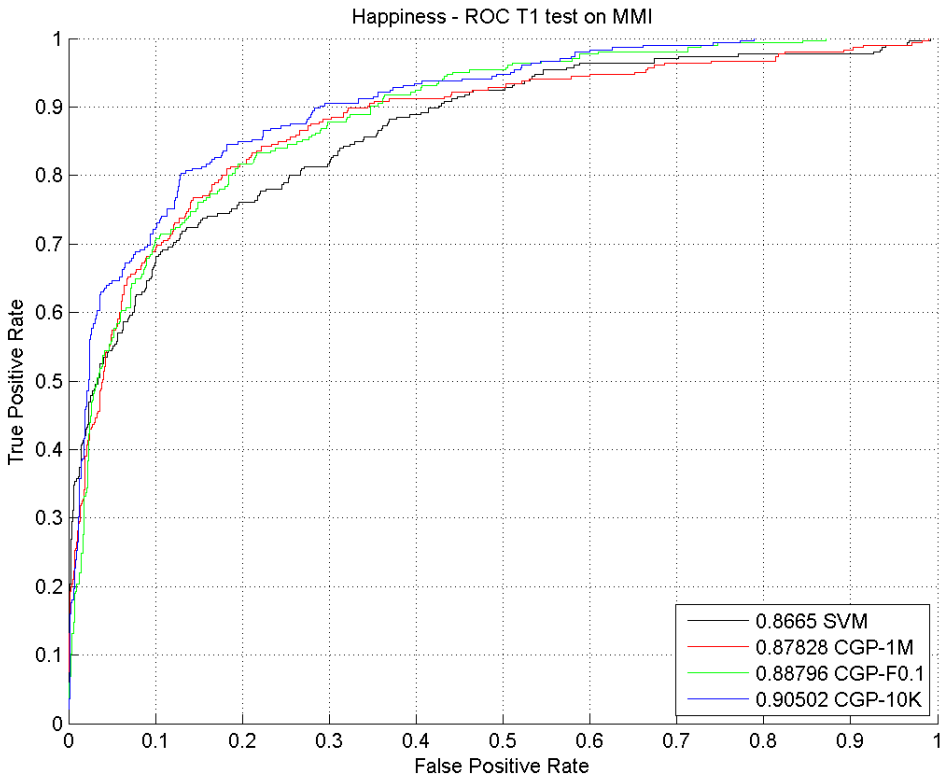


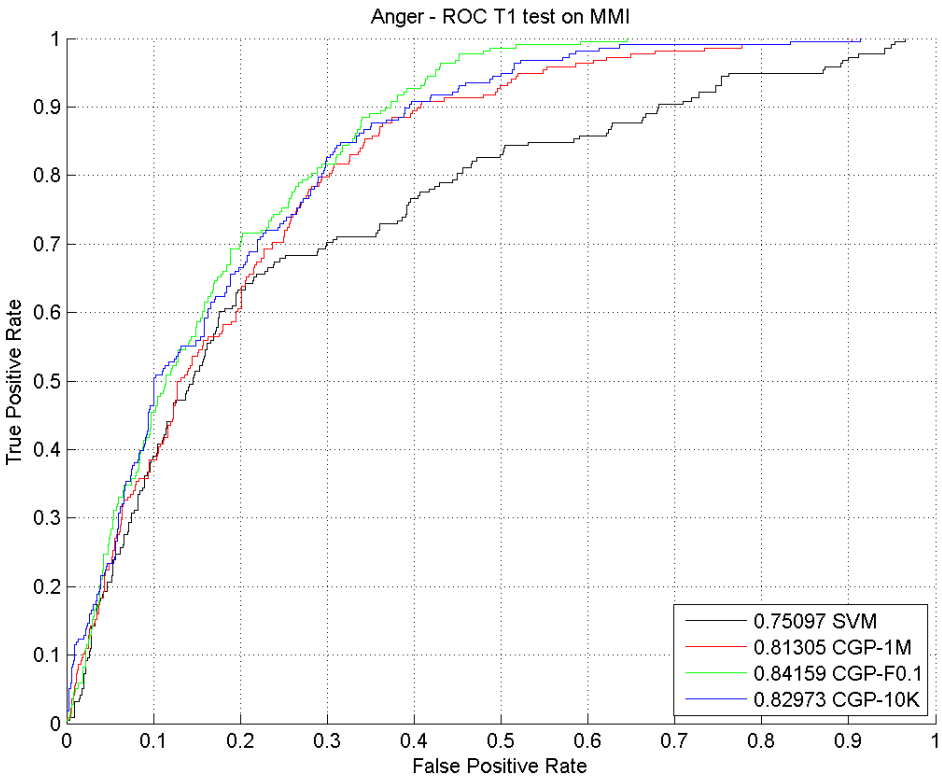
Figure 4.40: Comparison of ROC curve trained by MMI test with FG-NET (T1)

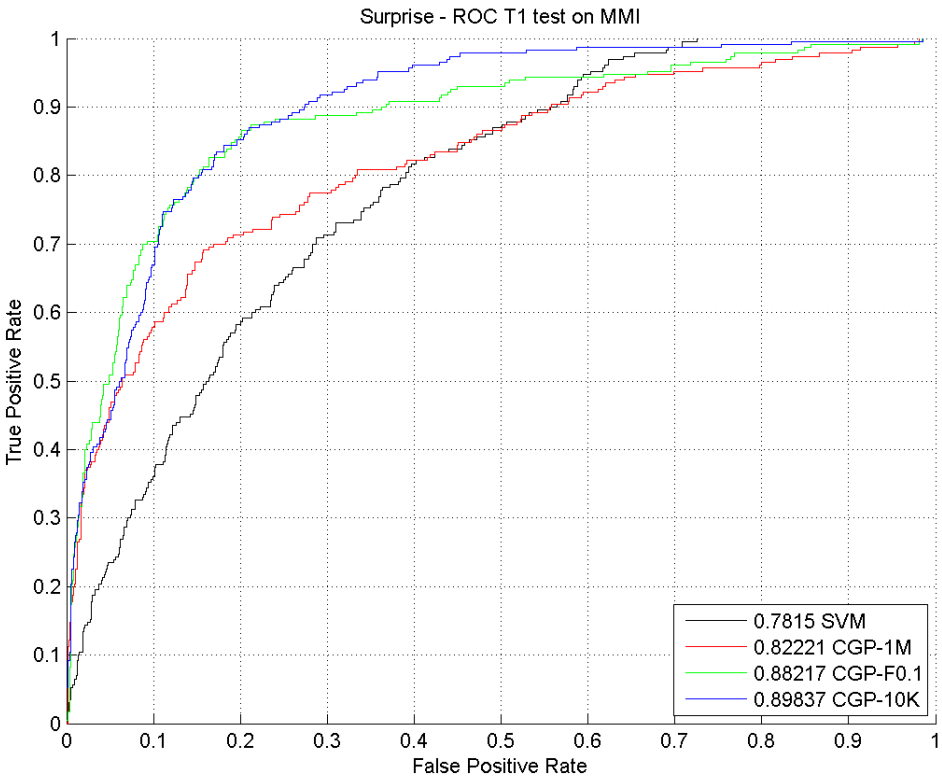
This figure shows the AUC of SVM and three CGP classifiers on testing FGNET expression data by 24 grids. The comparison shows: they have the similar performance on detecting happiness and

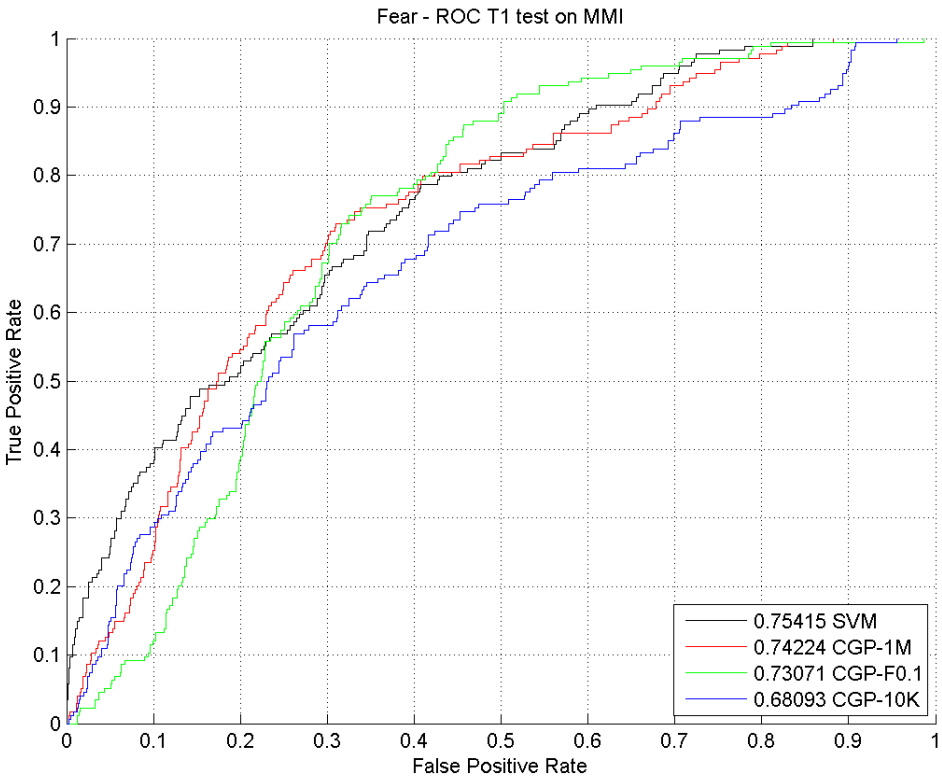
anger. In the figure of fear recognition, the fitness 0.1 and 10k generations CGP classifier have the larger area, but for practical use the point to the left should be selected, especially to the minority class. So the SVM classifier has good performance on sadness detection but poor in disgust recognition.

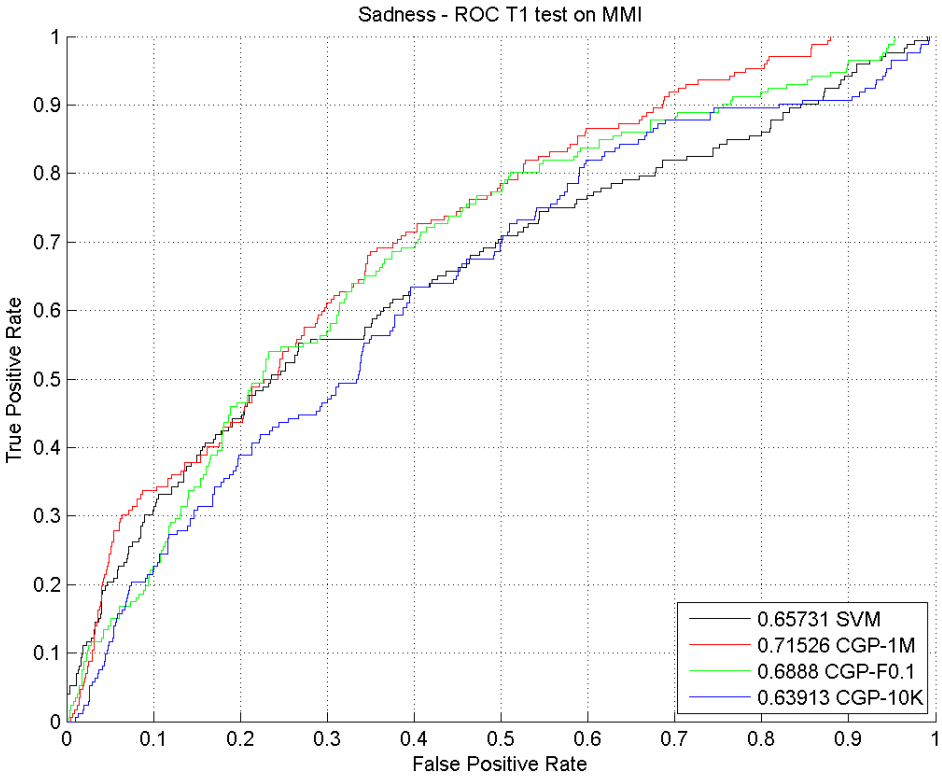
b) Trained by FGNET, test with MMI




Anger - ROC T1 test on MMI
True Positive Rate
False Positive Rate
0.75097 SVM
0.81305 CGP-1M
0.84159 CGP-F0.1
0.82973 CGP-10K


Surprise - ROC T1 test on MMI
True Positive Rate
False Positive Rate
0.7815 SVM
0.82221 CGP-1M
0.88217 CGP-F0.1
0.89837 CGP-10K


Fear - ROC T1 test on MMI
True Positive Rate
False Positive Rate
0.75415 SVM
0.74224 CGP-1M
0.73071 CGP-F0.1
0.68093 CGP-10K


Sadness - ROC T1 test on MMI
True Positive Rate
False Positive Rate
0.65731 SVM
0.71526 CGP-1M
0.6888 CGP-F0.1
0.63913 CGP-10K

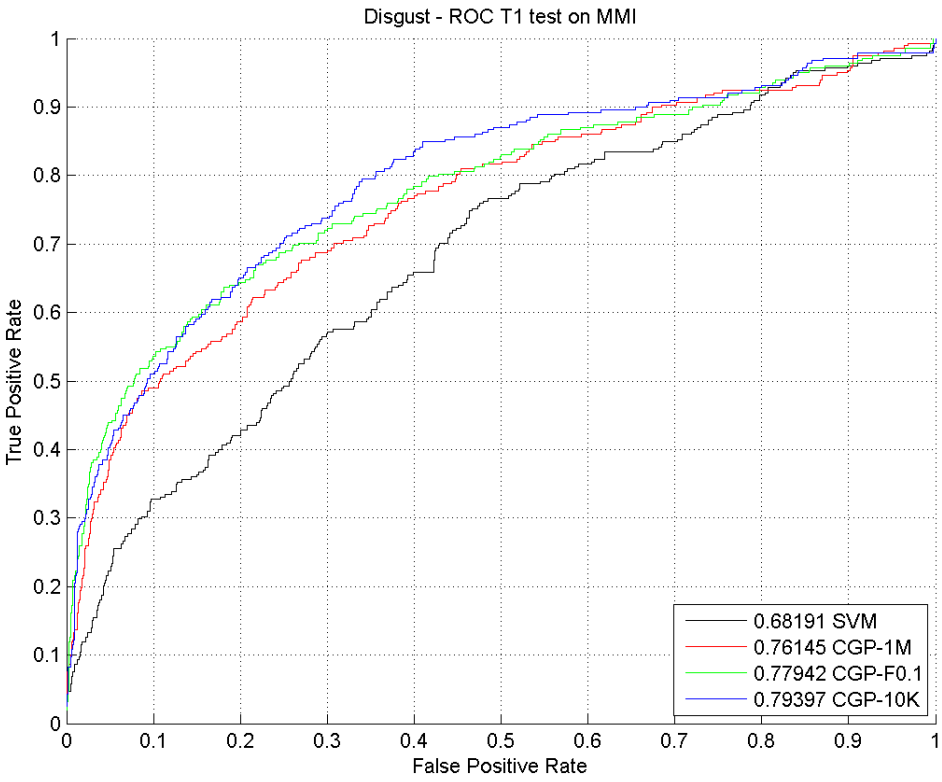
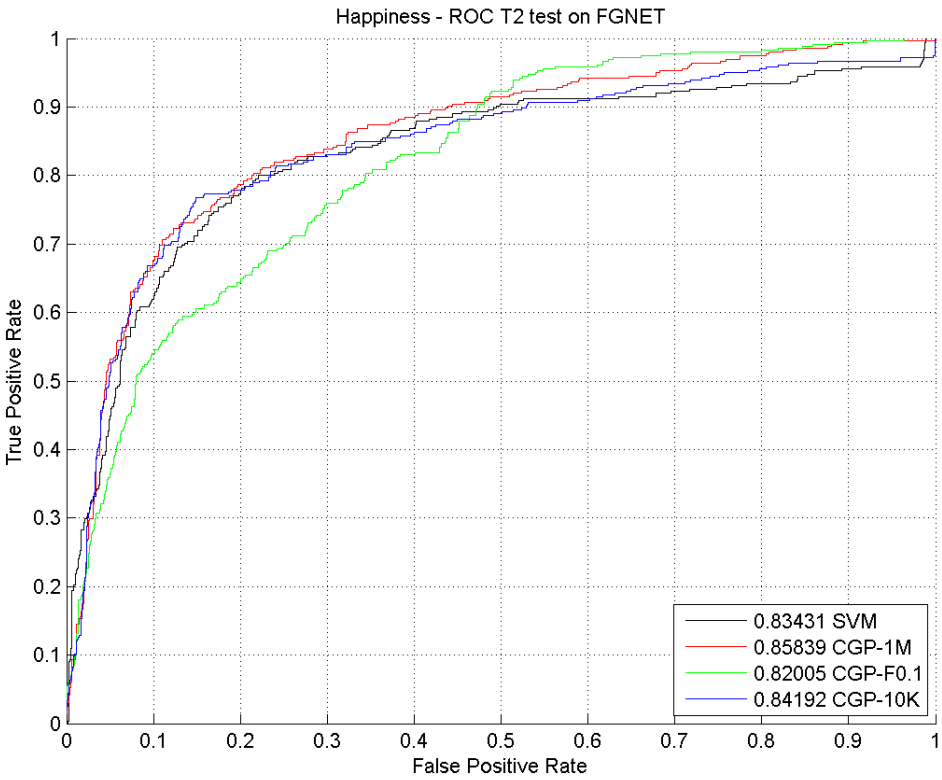


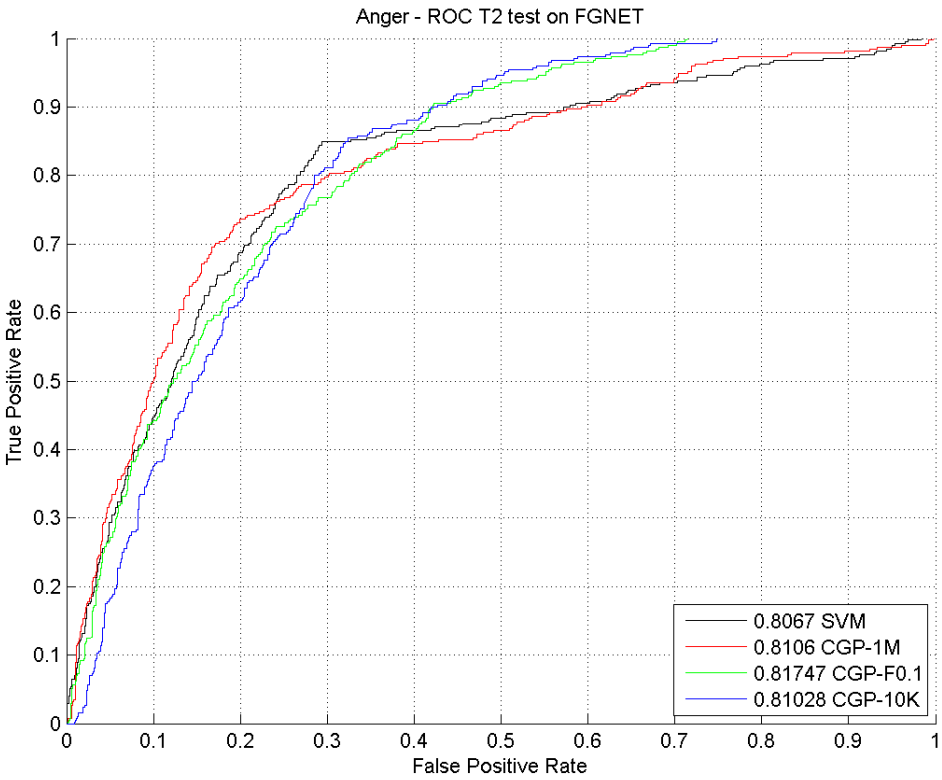
Figure 4.41: Comparison of ROC curve trained by FG-NET test with MMI (T1)

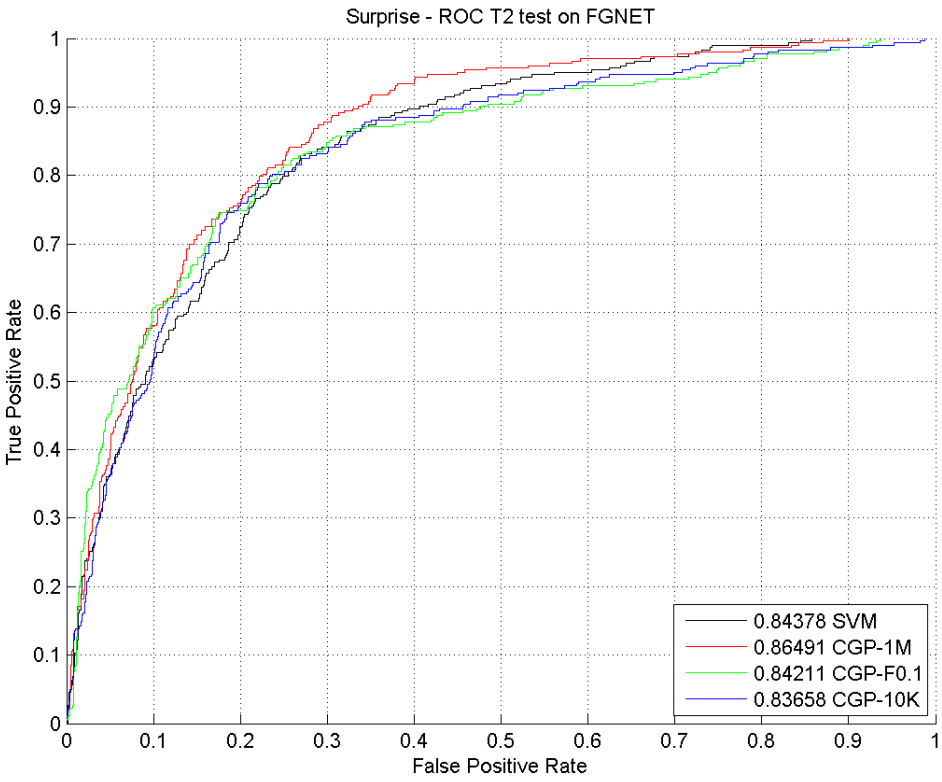
This experiment is training the classifier with FGNET database then test the classifier with MMI expressions. CGP classifiers are better than SVM on surprise and disgust recognition. The 10K generations CGP classifier is better except on fear and sadness which has too few samples in FG-NET.

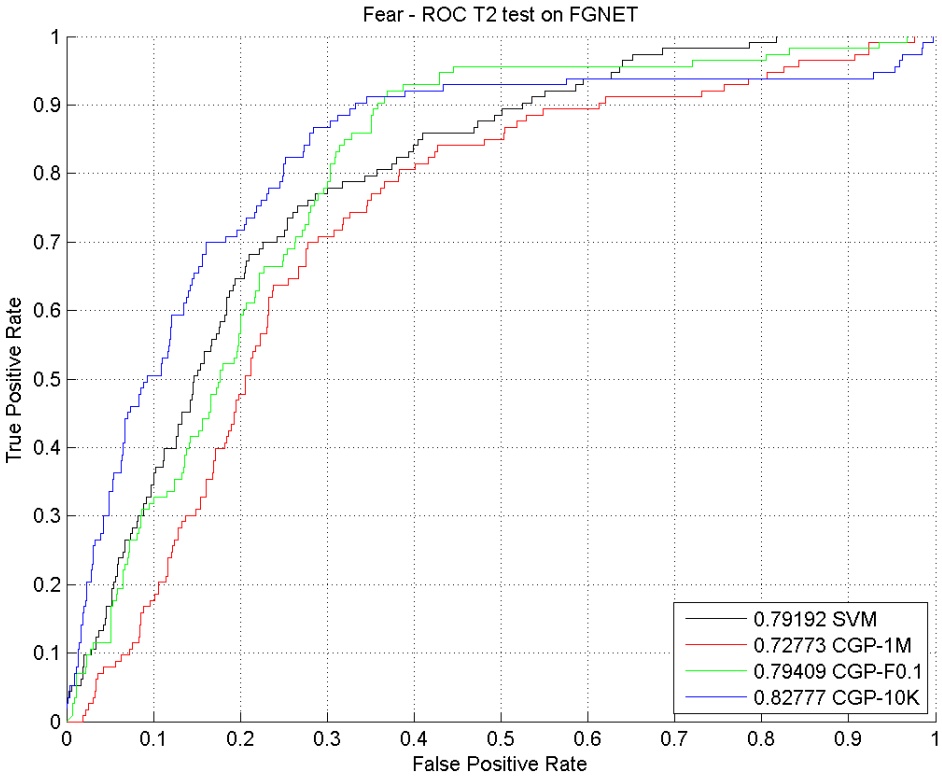
4.6.3.2 AUC Graph of 48 Grids (T2)

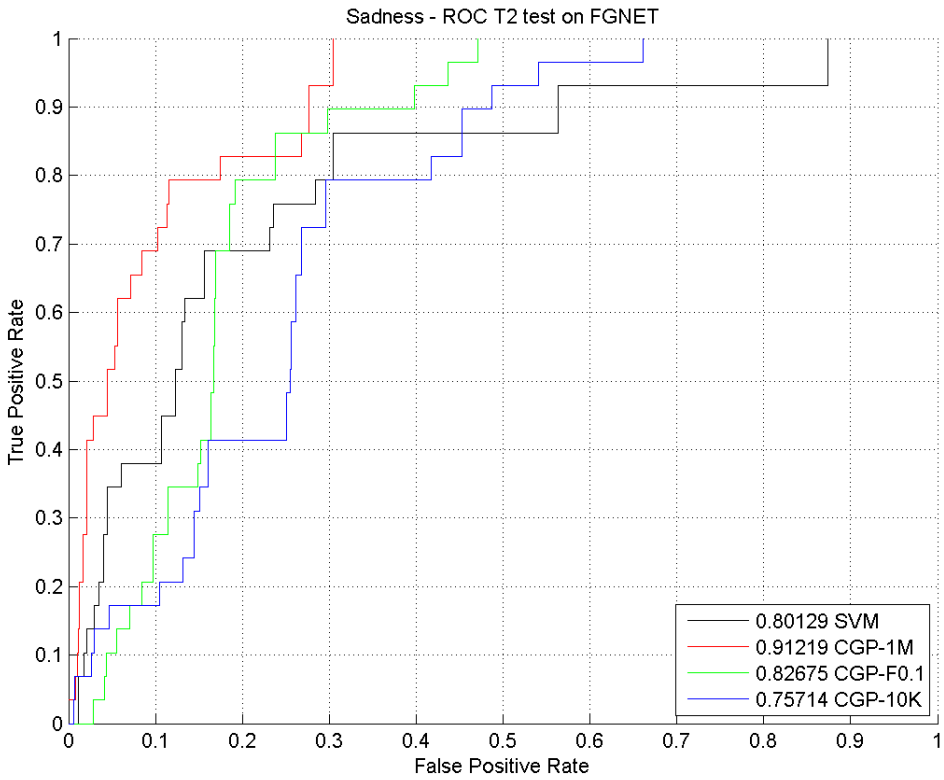
a) Trained by MMI, test with FGNET






Surprise - ROC T2 test on FGNET
True Positive Rate
False Positive Rate
0.84378 SVM
0.86491 CGP-1M
0.84211 CGP-F0.1
0.83658 CGP-10K


Fear - ROC T2 test on FGNET
True Positive Rate
False Positive Rate
0.79192 SVM
0.72773 CGP-1M
0.79409 CGP-F0.1
0.82777 CGP-10K



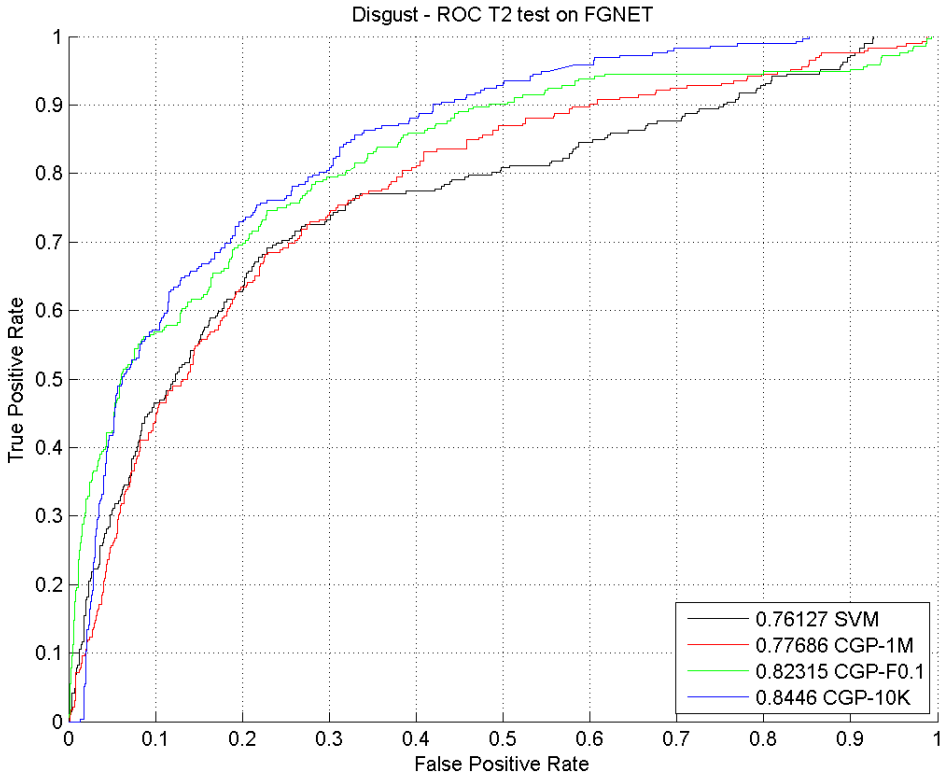
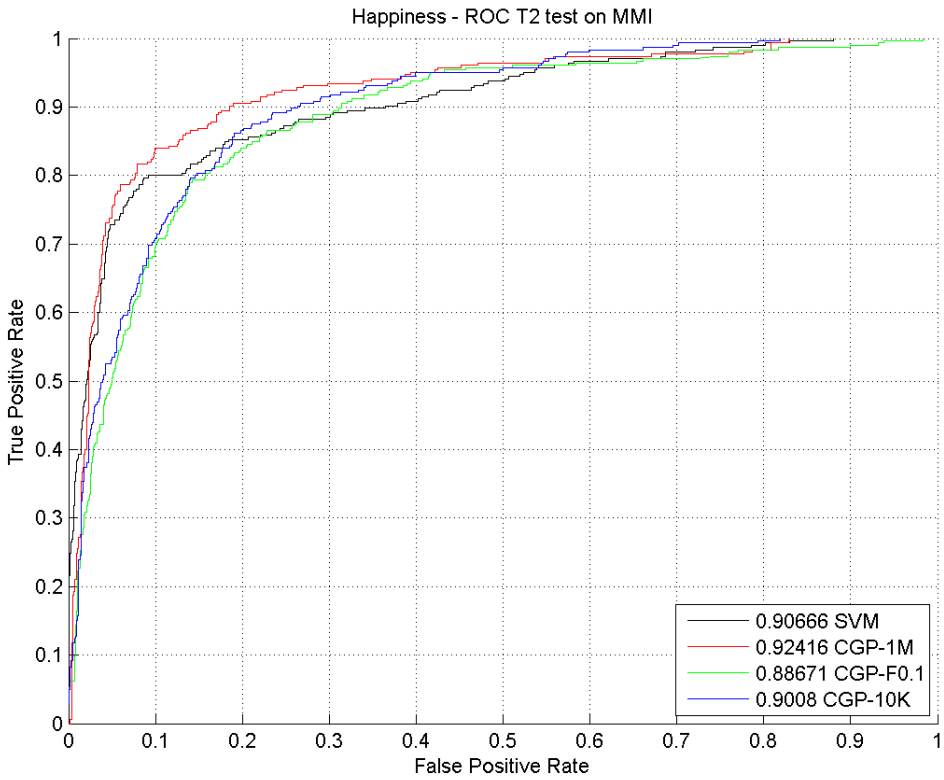


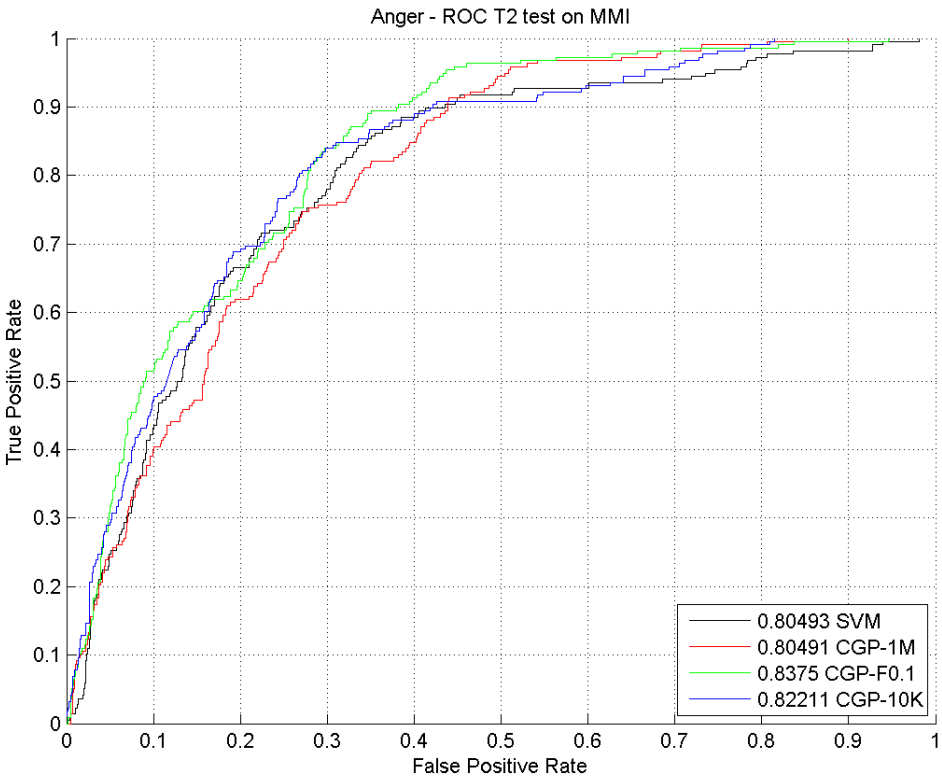
Figure 4.42: Comparison of ROC curve trained by MMI test with FG-NET (T2)

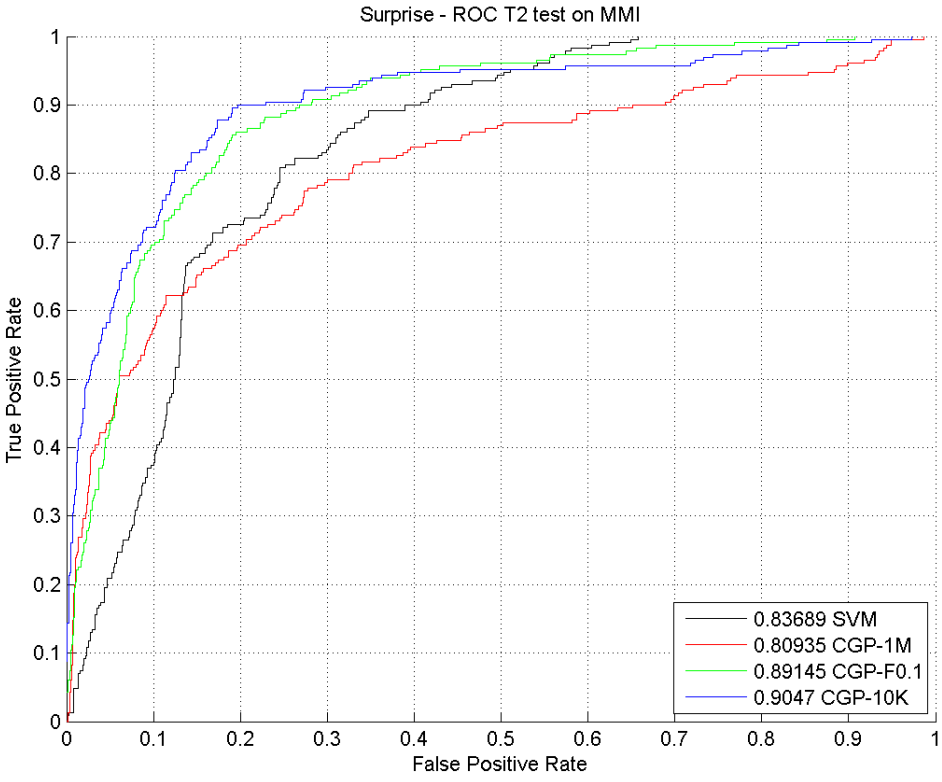
This experiment divides the face image into 48 grids. The CGP classifier after one million generations' evolution has better performance on sadness and angry detection, but worse at fear

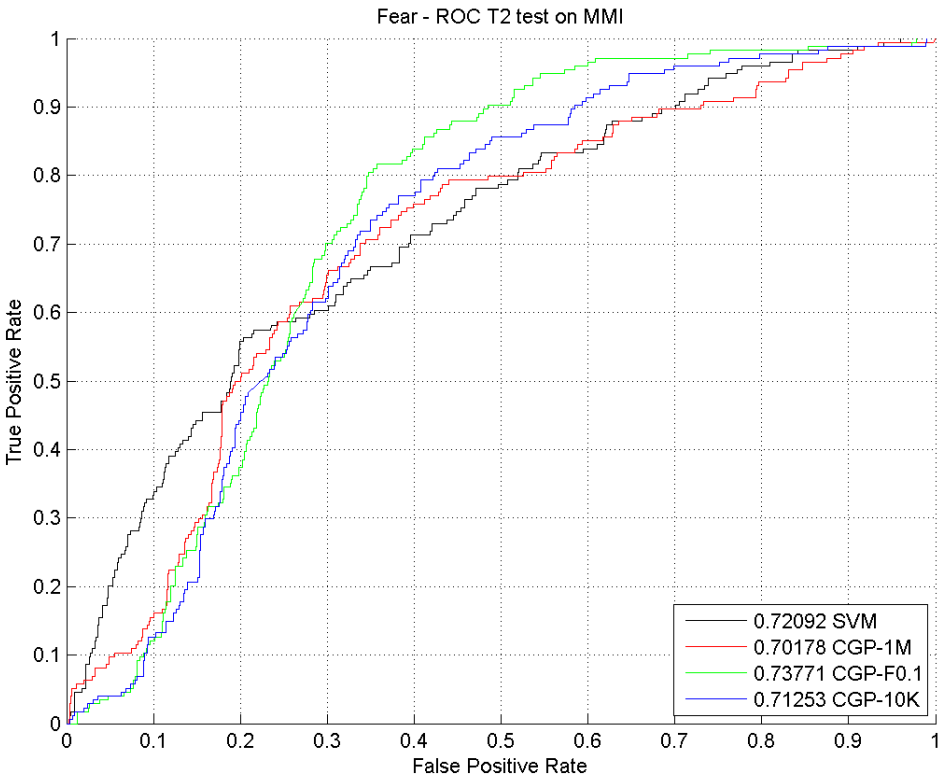
and disgust classification. The 10K generation CGP classifier performs well at fear detection but lose in sadness recognition. The SVM classifier is in the middle of CGP classifiers, it does not have the best or worst result on any expression in this experiment.

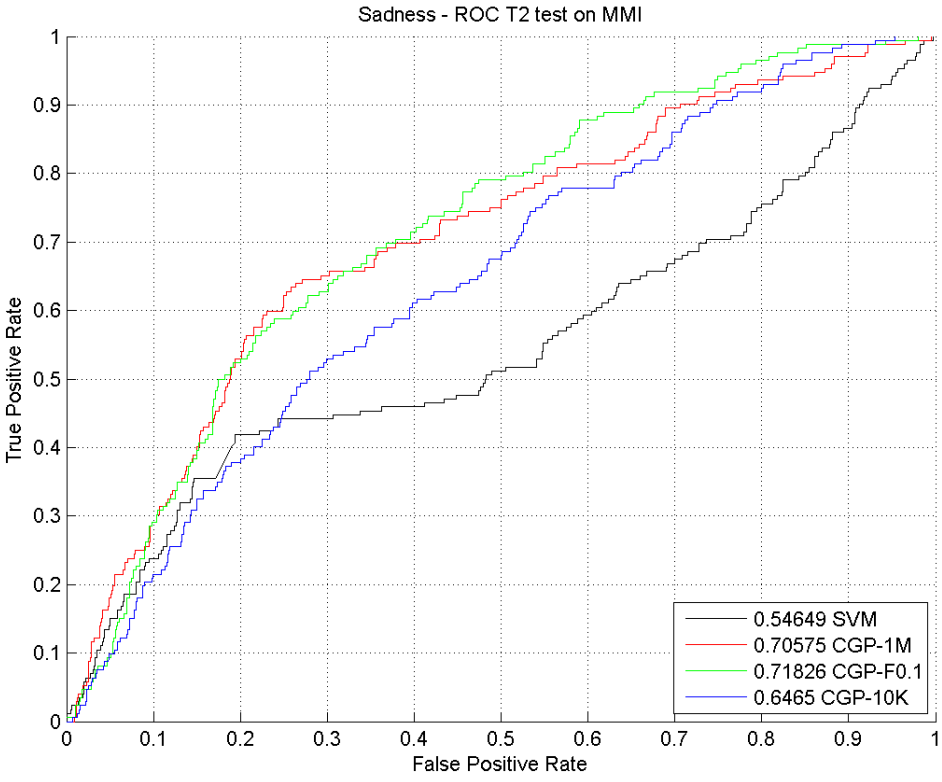
b) Trained by FGNET, test with MMI




Anger - ROC T2 test on MMI
True Positive Rate
False Positive Rate
0.80493 SVM
0.80491 CGP-1M
0.8375 CGP-F0.1
0.82211 CGP-10K


Surprise - ROC T2 test on MMI
True Positive Rate
False Positive Rate
0.83689 SVM
0.80935 CGP-1M
0.89145 CGP-F0.1
0.9047 CGP-10K


Fear - ROC T2 test on MMI
True Positive Rate
False Positive Rate
0.72092 SVM
0.70178 CGP-1M
0.73771 CGP-F0.1
0.71253 CGP-10K


Sadness - ROC T2 test on MMI
True Positive Rate
False Positive Rate
0.54649 SVM
0.70575 CGP-1M
0.71826 CGP-F0.1
0.6465 CGP-10K

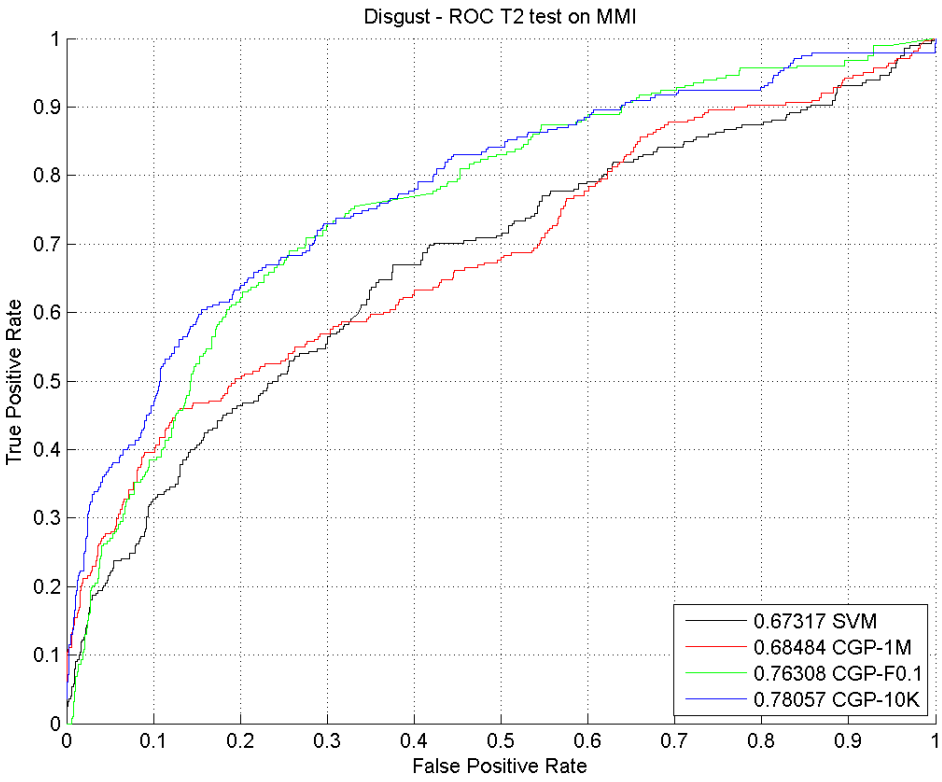
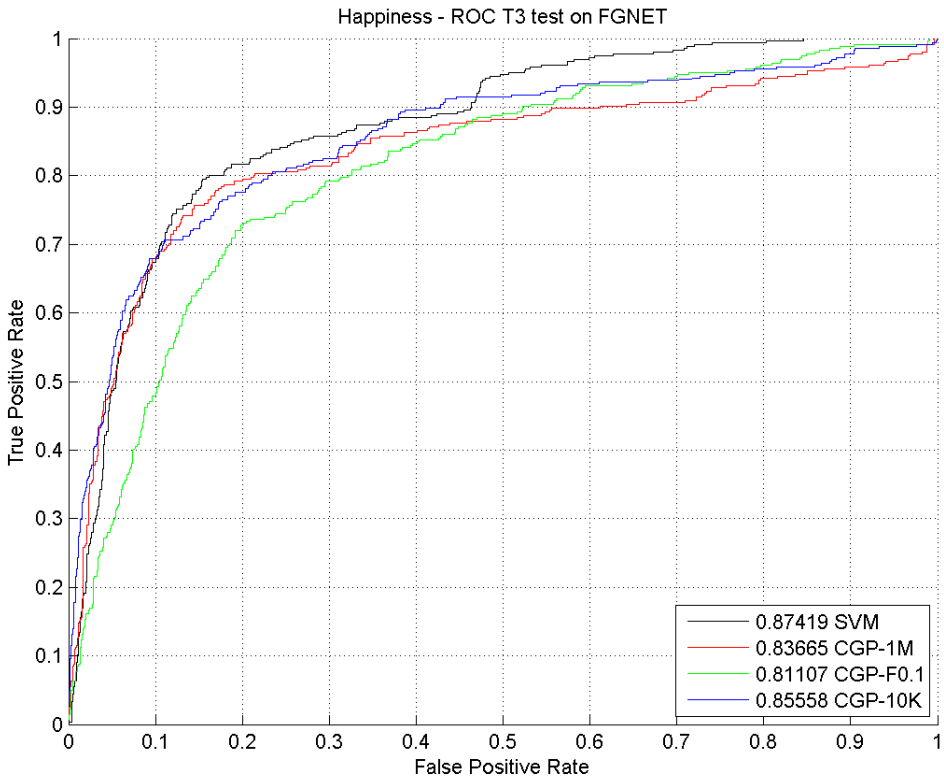


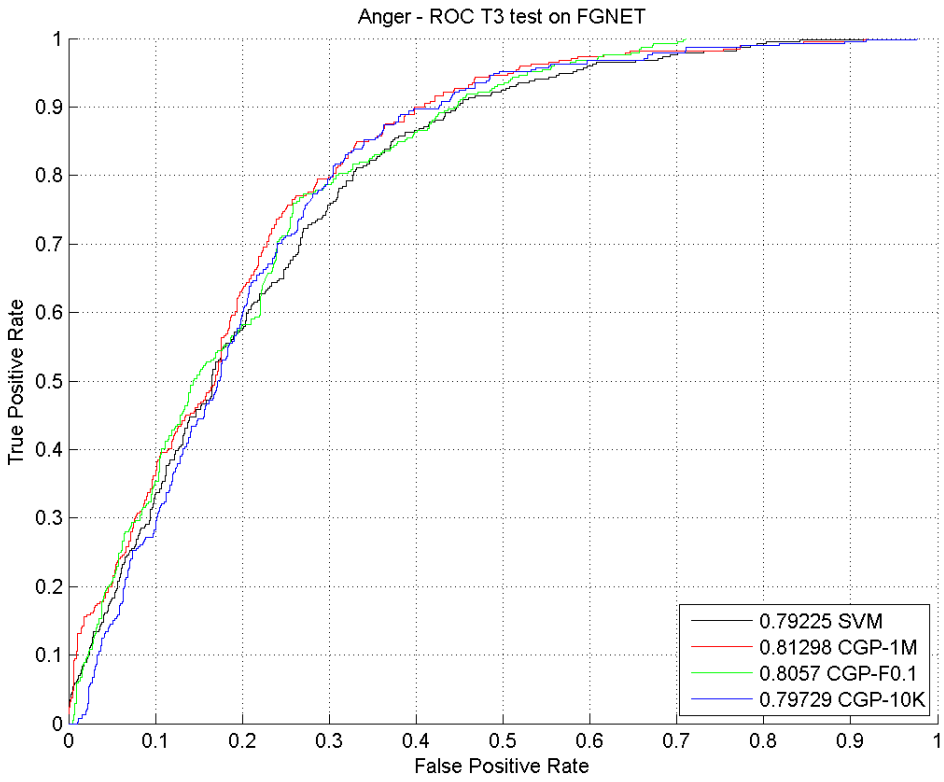
Figure 4.43: Comparison of ROC curve trained by FG-NET test with MMI (T2)

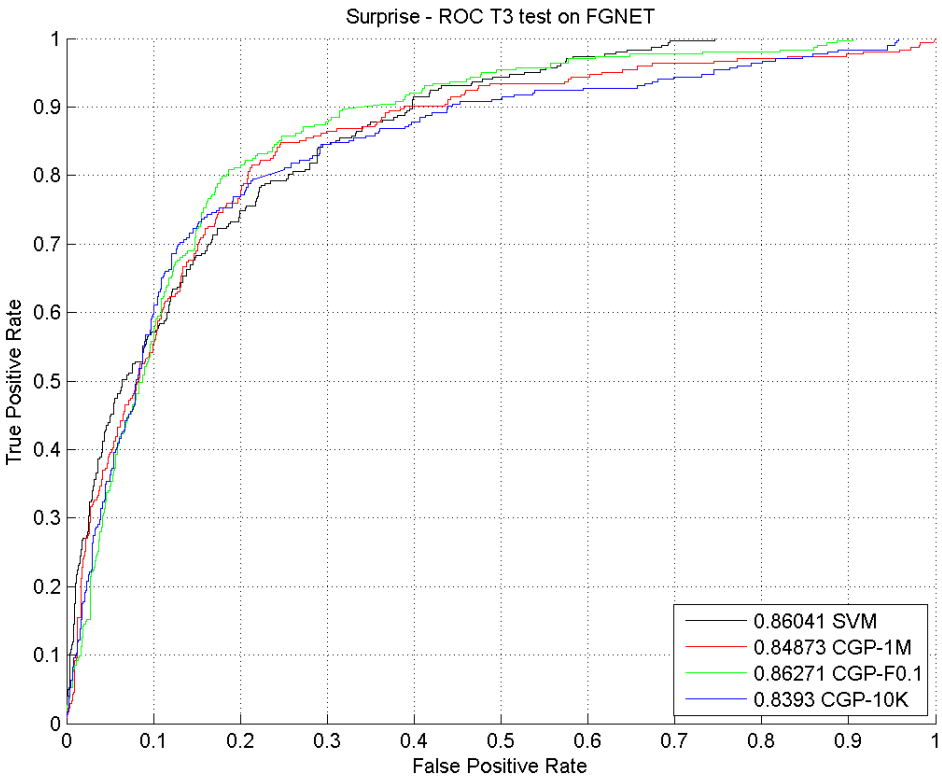
In this experiment, SVM is better on fear detection and all the CGP classifiers are concave at the left part. SVM also performs well on happiness but is the worst in surprise, sadness and disgust. Red line is the one million generation CGP classifier. It has very good performance on happy but bad in surprise and disgust.

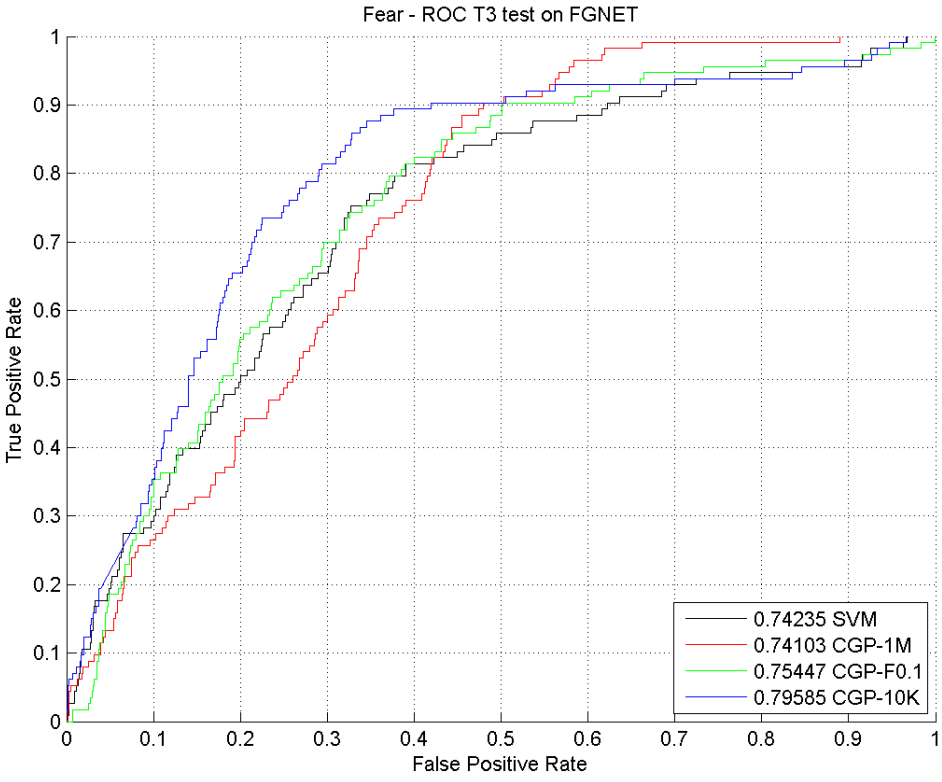
4.6.3.3 AUC Graph of 96 Grids (T3)

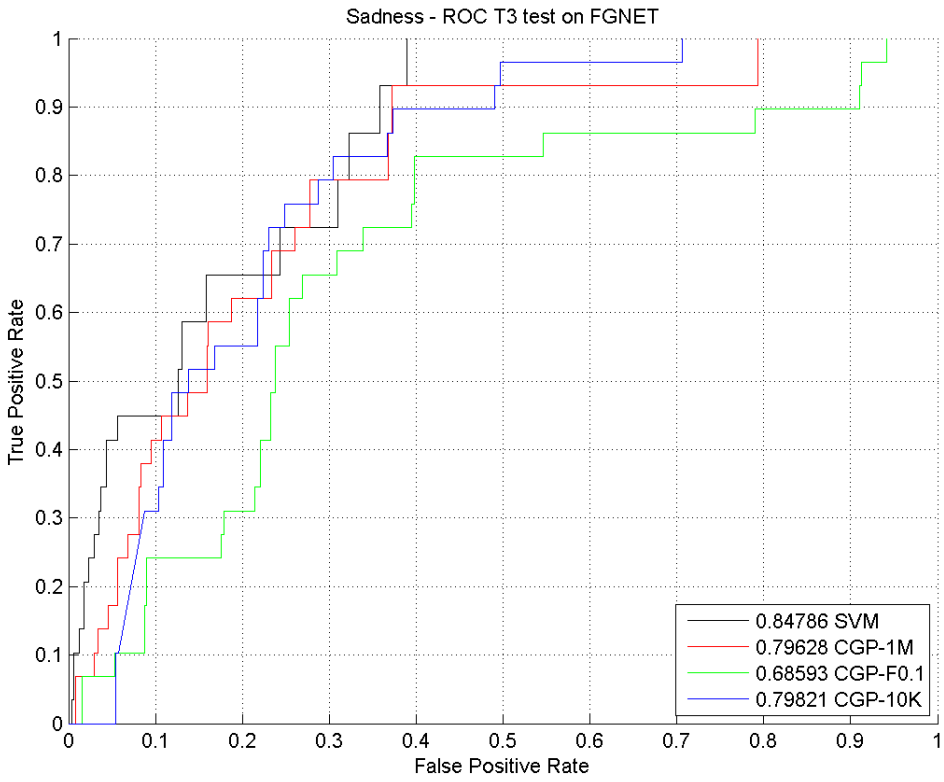
a) Trained by MMI, test with FGNET






Surprise - ROC T3 test on FGNET
True Positive Rate
False Positive Rate
0.86041 SVM
0.84873 CGP-1M
0.86271 CGP-F0.1
0.8393 CGP-10K


Fear - ROC T3 test on FGNET
True Positive Rate
False Positive Rate
0.74235 SVM
0.74103 CGP-1M
0.75447 CGP-F0.1
0.79585 CGP-10K



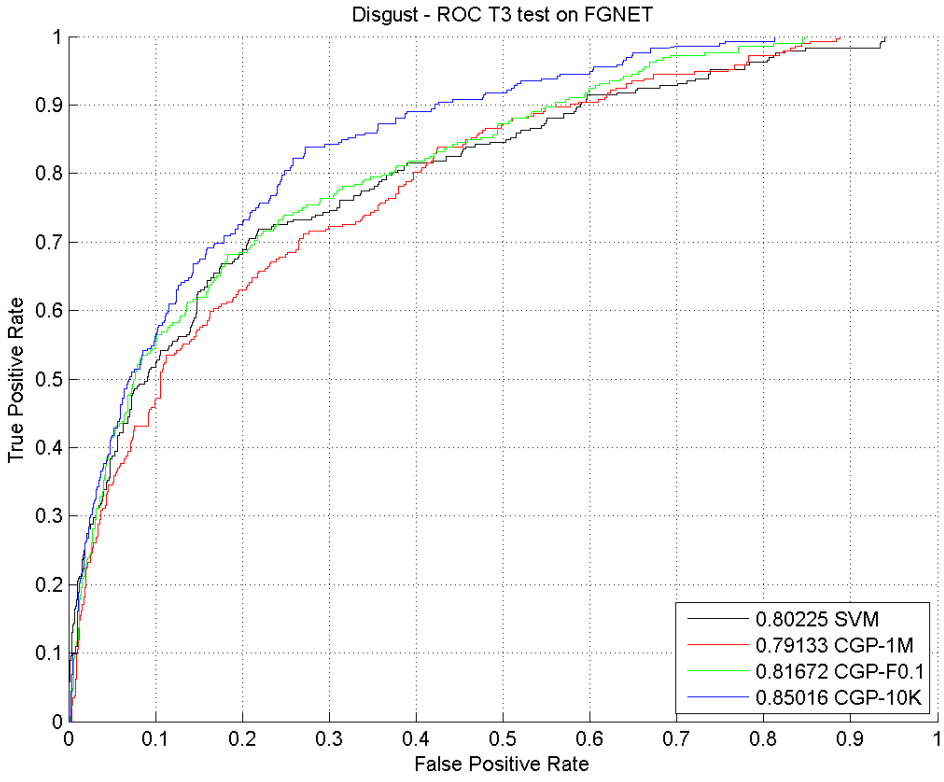
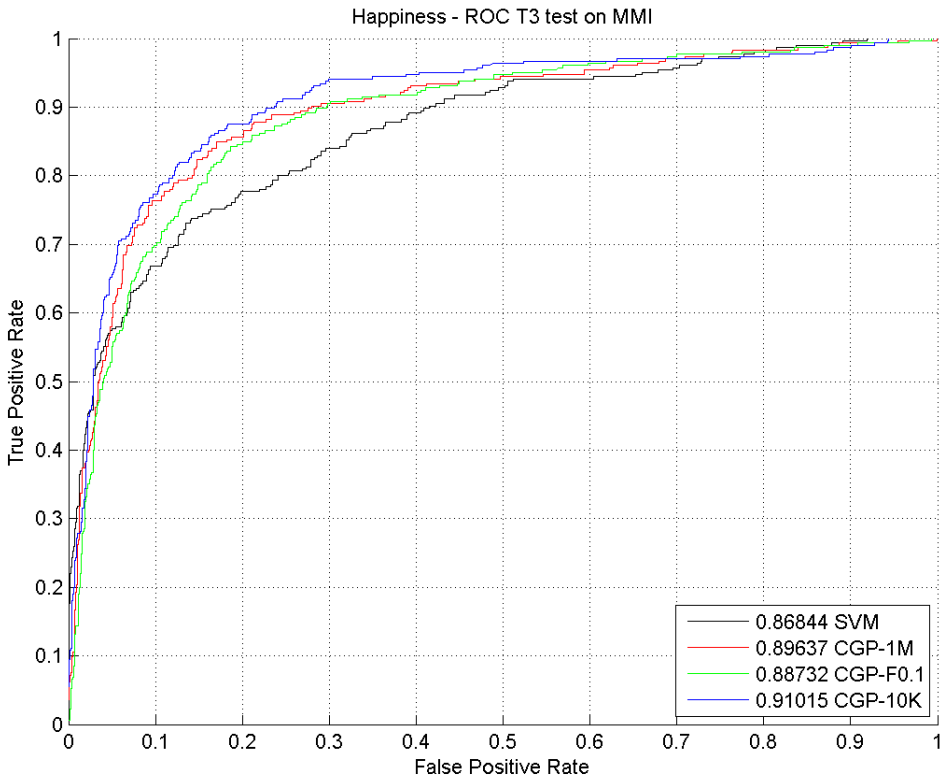


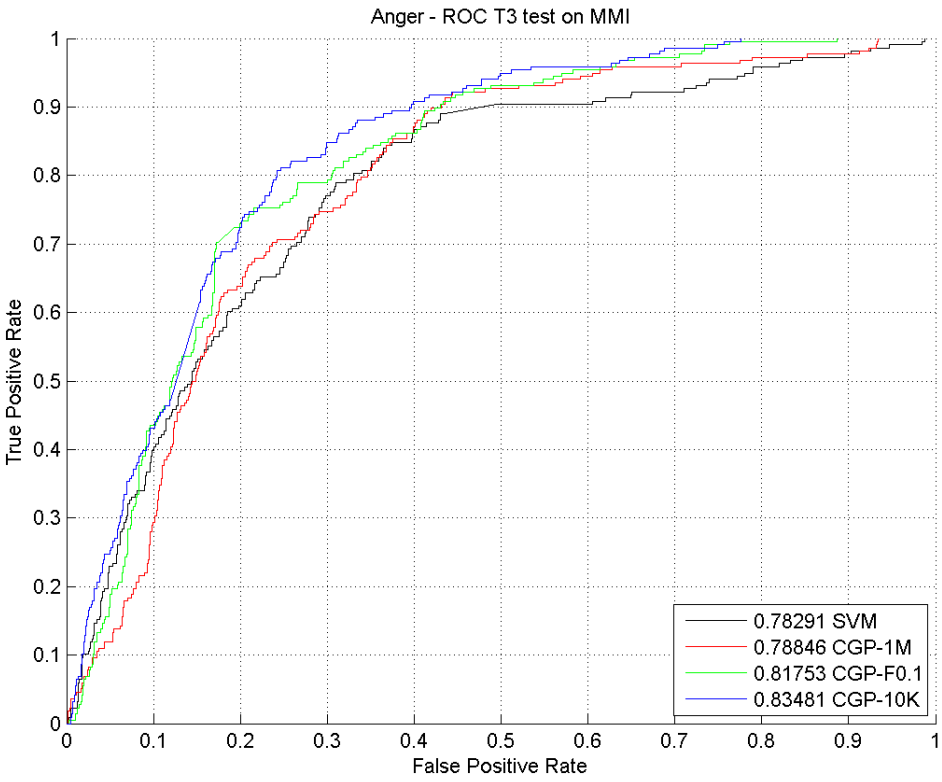
Figure 4.44: Comparison of ROC curve trained by MMI test with FG-NET (T3)

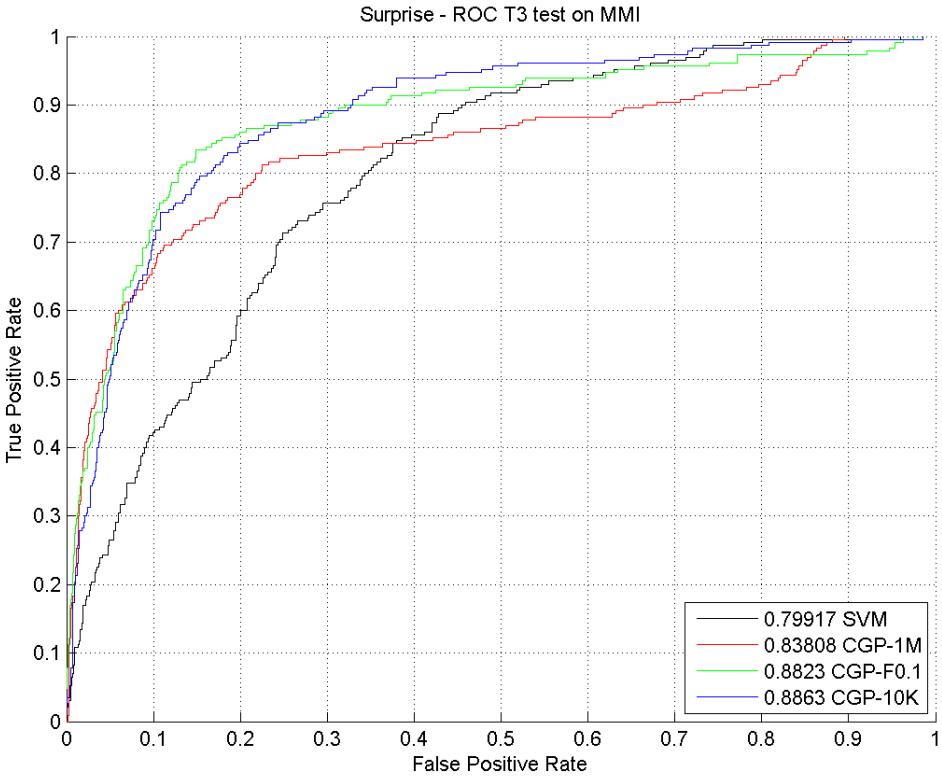
This experiment increases the girds number further to 96. The corresponding feature length is 192 (each grid has x and y velocity). This time the CGP classifier with 10K generation evolution

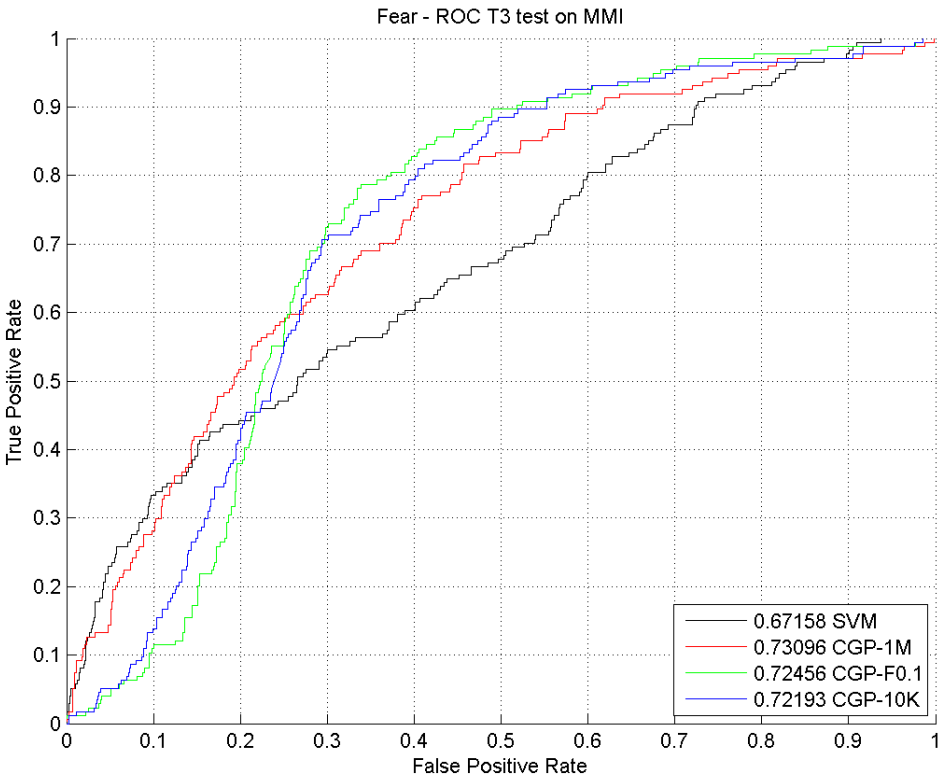
always performs well. Red line is weak on fear expression and green line on sadness. Other classifiers do not have too many differences.

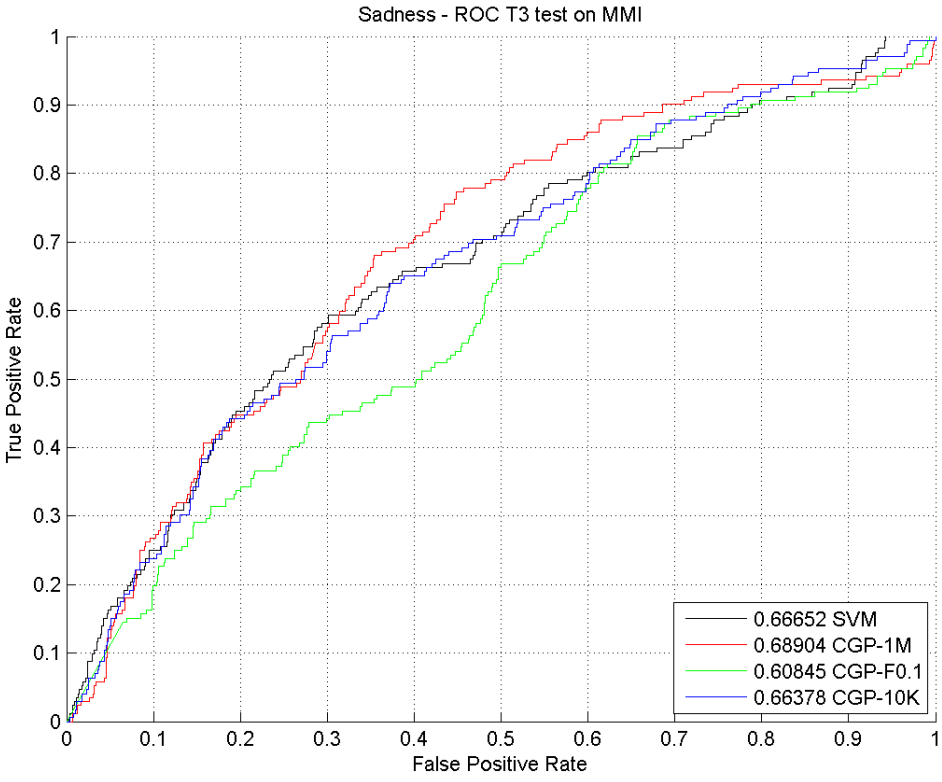
b) Trained by FGNET, test with MMI




Anger - ROC T3 test on MMI
True Positive Rate
False Positive Rate
0.78291 SVM
0.78846 CGP-1M
0.81753 CGP-F0.1
0.83481 CGP-10K


Surprise - ROC T3 test on MMI
True Positive Rate
False Positive Rate
0.79917 SVM
0.83808 CGP-1M
0.8823 CGP-F0.1
0.8863 CGP-10K


Fear - ROC T3 test on MMI
True Positive Rate
False Positive Rate
0.67158 SVM
0.73096 CGP-1M
0.72456 CGP-F0.1
0.72193 CGP-10K


Sadness - ROC T3 test on MMI
True Positive Rate
False Positive Rate
0.66652 SVM
0.68904 CGP-1M
0.60845 CGP-F0.1
0.66378 CGP-10K

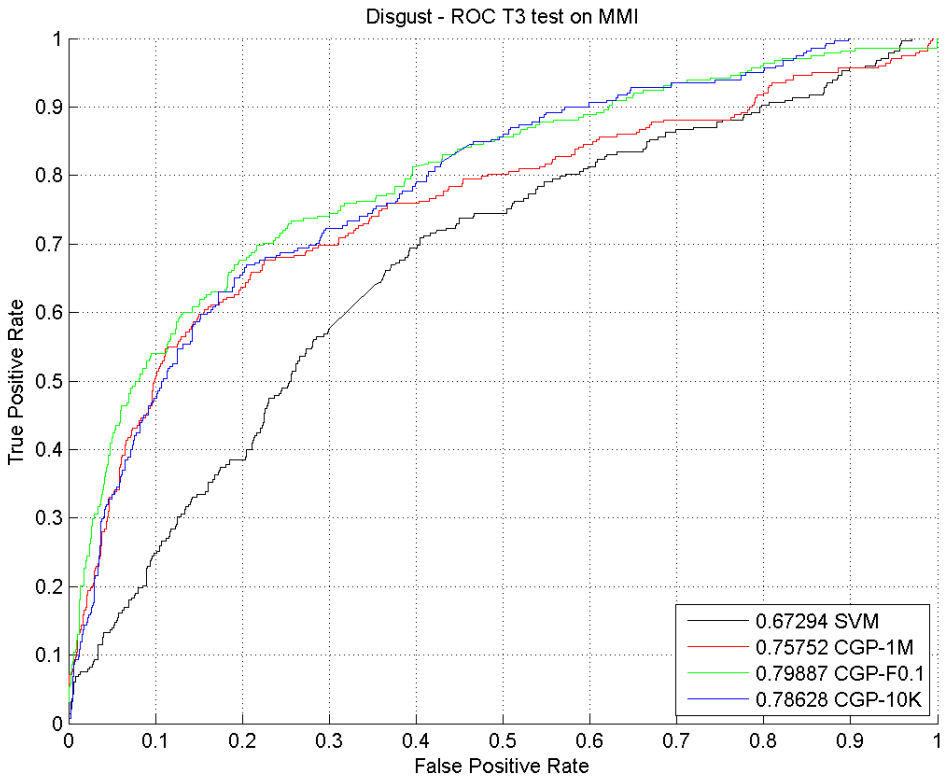


Figure 4.45: Comparison of ROC curve trained by FG-NET test with MMI (T3)

SVM classifier has the smallest AUC in this set of experiments, although it is not bad on angry, fear and sadness expression recognition. The red line and blue one have the better overall performance, but green and blue cave in again at fear expression classification.

4.6.3.4 Results Analysis

Table 4.10: Comparison of AUC between CGP and SVM classifier (T1)

T1	Train MMI Test FG-NET				Train FG-NET Test MMI			
	SVM	CGP-M	CGP-F0.1	CGP-10K	SVM	CGP-M	CGP-F0.1	CGP-10K
Happiness	**0.8709**	*0.8388*	0.8528	0.8514	*0.8665*	0.8783	0.888	**0.905**
Anger	*0.775*	0.7989	**0.8025**	0.8023	*0.751*	0.8131	**0.8416**	0.8297
Surprise	*0.8467*	0.8489	**0.8667**	0.8662	*0.7815*	0.8222	0.8822	**0.8984**
Fear	0.7515	*0.73*	0.7906	**0.7993**	**0.7542**	0.7422	0.7307	*0.6809*
Sadness	**0.8942**	*0.7876*	0.7882	0.7925	0.6573	**0.7153**	0.6888	*0.6391*
Disgust	*0.7307*	0.8121	0.8145	**0.8464**	*0.6819*	0.7615	0.7794	**0.794**

Table 4.11: Comparison of AUC between CGP and SVM classifier (T2)

T2	Train MMI Test FG-NET				Train FG-NET Test MMI			
	SVM	CGP-M	CGP-F0.1	CGP-10K	SVM	CGP-M	CGP-F0.1	CGP-10K
Happiness	0.8343	**0.8584**	*0.82*	0.8419	0.9067	**0.9242**	*0.8867*	0.9008
Anger	*0.8067*	0.8106	**0.8175**	0.8103	0.80493	*0.80491*	**0.8375**	0.8221
Surprise	0.8438	**0.8649**	0.8421	*0.8366*	0.8369	*0.8093*	0.8915	**0.9047**
Fear	0.7919	*0.7277*	0.7941	**0.8278**	0.7209	*0.7018*	**0.7377**	0.7125
Sadness	0.8013	**0.9122**	0.8268	*0.7571*	*0.5465*	0.7057	**0.7183**	0.6465
Disgust	*0.7613*	0.7769	0.8231	**0.8446**	*0.6732*	0.6848	0.7631	**0.7806**

Table 4.12: Comparison of AUC between CGP and SVM classifier (T3)

T3	Train MMI Test FG-NET				Train FG-NET Test MMI			
	SVM	CGP-M	CGP-F0.1	CGP-10K	SVM	CGP-M	CGP-F0.1	CGP-10K
Happiness	**0.8742**	0.8367	*0.8111*	0.8556	*0.8684*	0.8964	0.8873	**0.9102**
Anger	*0.7922*	**0.813**	0.8057	0.7973	*0.7829*	0.7885	0.8175	**0.8348**
Surprise	0.8604	0.8487	**0.8627**	*0.8393*	*0.7992*	0.8381	0.8823	**0.8863**
Fear	0.7424	*0.741*	0.7545	**0.7958**	*0.6716*	**0.731**	0.7246	0.7219
Sadness	**0.8479**	0.7963	*0.6859*	0.7982	0.6665	**0.689**	*0.6084*	0.6638
Disgust	0.8022	*0.7913*	0.8167	**0.8502**	*0.6729*	0.7575	**0.7989**	0.7863

Tables 4.10-4.12 show the results of SVM and three different CGP classifiers on AUC of expression recognition. The numbers in bold indicate they the highest among the four classifier on that experiment. The numbers in Italic type are the lowest AUC in the four classifiers. Then another comparison can be obtained.

Table 4.13: Times of the classifier have the best/worst AUC

Times of	SVM	CGP-M	CGP-F0.1	CGP-10K
Best	5	8	9	14
Worst	17	9	5	5
Compare with SVM	Higher	23	27	24
	Lower	13	9	12

After the comparison of 36 plots, in five figures the four classifiers do not have much difference. The SVM classifier in 1/5 time has the worst performance, and leads the AUC four times. The CGP classifier with 10K training generations is slightly better in the series of experiments. But there is no single classifier has the overall best performance, the most common case is strong in some expressions but weak than others in another expression. For fear and sadness expression, the number of samples is too small, the SVM and CGP with one million generation evolution have better performance.

Another interesting point is the imbalanced training data makes the false positive rate easily low and the true positive is difficult to achieve a high rate. In practical application, the low pass or

high pass of the positive samples is fixed. So the opposite side is not important, but still considered in the AUC fitness function. Although with the characteristic of AUC, the threshold can be select freely on the curve, it is still possible to evolve a better classifier if the low pass or high pass is predefined.–

The accuracy is still very low except happiness detection. Obvious reasons are certainly the ultra-imbalanced data set and the differences between the two databases. The worst imbalanced case is 29:1444 of sadness in FGNET database. Another reason is: the feature is taken from frames; each frame is one sample for classification. The rising part of one expression is usually 10-20 frames period. The accuracy is counted by each frame sample from one expression procedure. This means only correctly classify each frame during the expression can achieve a good recognition rate, which is a tough task. The very beginning of expression (first few frames) has tiny movement and difficult to distinguish. The ROC curve on a test data is very difficult to be perfect with the start of an expression is counted in the data set. In the practical used in an expression recognition system, the low pass threshold and a stack of positive respond in a short period are capable to do the job.

4.7 Implementation

The binary classifiers together with face detection technique are used for building the facial expression recognition system. The system implemented by C++ with OpenCV library[139] in visual studio 2010. The most trainings of CGP classifier are done in Unix with Intel compiler[144] on the high performance computer, N8 HPC[145]. Viola and Jones' method is used for detecting human face. Then the flow features are extracted within the face area and passed to six binary classifiers at the same time. The binary classifiers give the results separately on its expression.

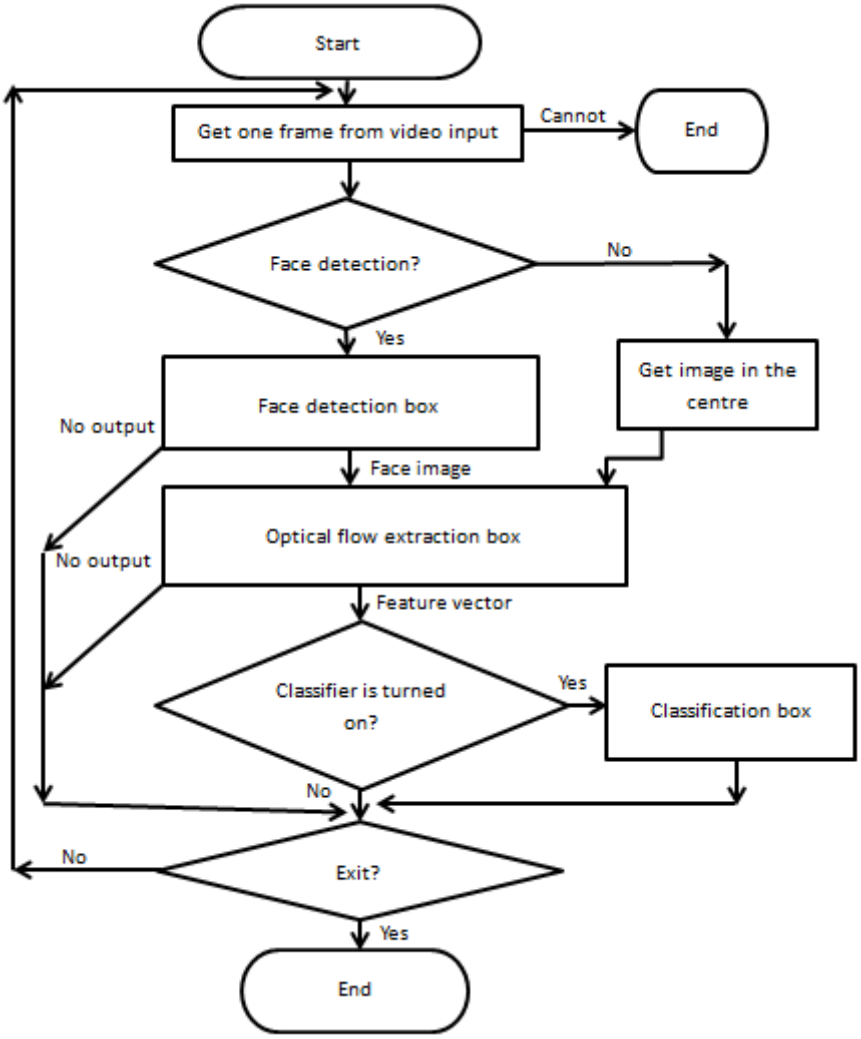


Figure 4.46: Flow chart of the expression recognition system

4.7.1 Input

The system works with a video stream which can feed from camera or a video file. Then the option to use face detection or not can be selected and switching by user. If turn on the face detection, the face area will be extracted in the picture by Viola and Jones' algorithm (introduced in Chapter 2). The resolution of the video affects the face detection speed, a scale factor can be set when input the video to resize the picture to process. If the face detection is turned down for the case that face cannot be detected stably, a fixed square area in the centre of the picture is applied the optical flow feature extraction and then to classifier. The centre area is a square with 1/3 length of the shorter side of the picture.

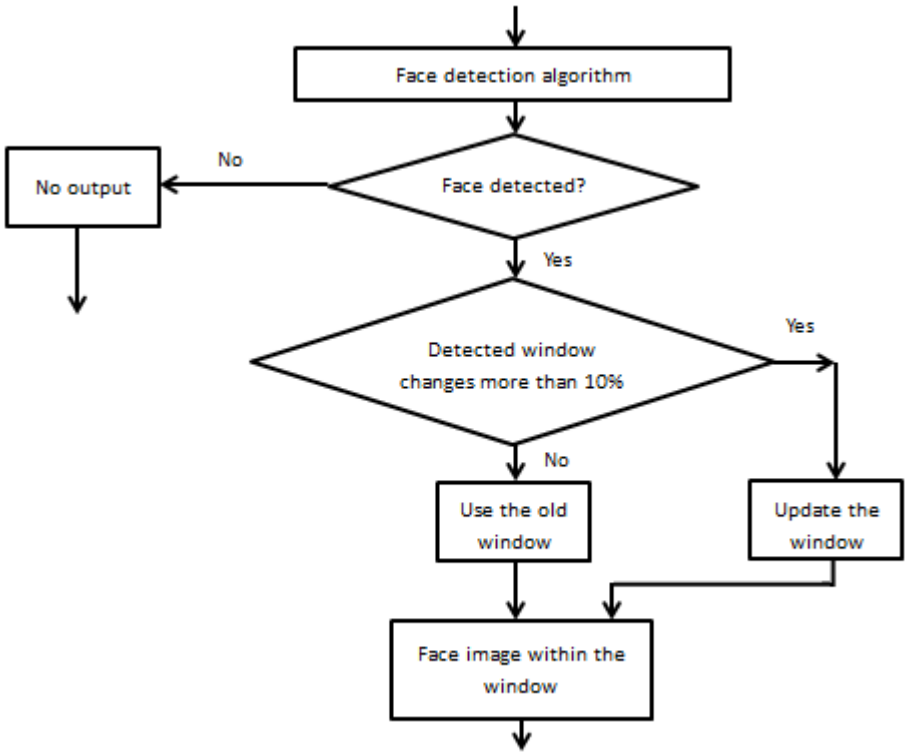


Figure 4.47: Flow chart inside face detection box

4.7.2 Feature Extraction

Image extracted from last step each frame is stored in an image array and the size of the array is 5 by default. When the array is full, the dense optical flow will be extracted between the first element and the last. Then the flow will be divided into 24 grids and the average flow of each grid forms the final feature vector as written in Chapter 3. The next frame a new image will push into the array, and all the elements in the array move one position and the first element will be deleted.

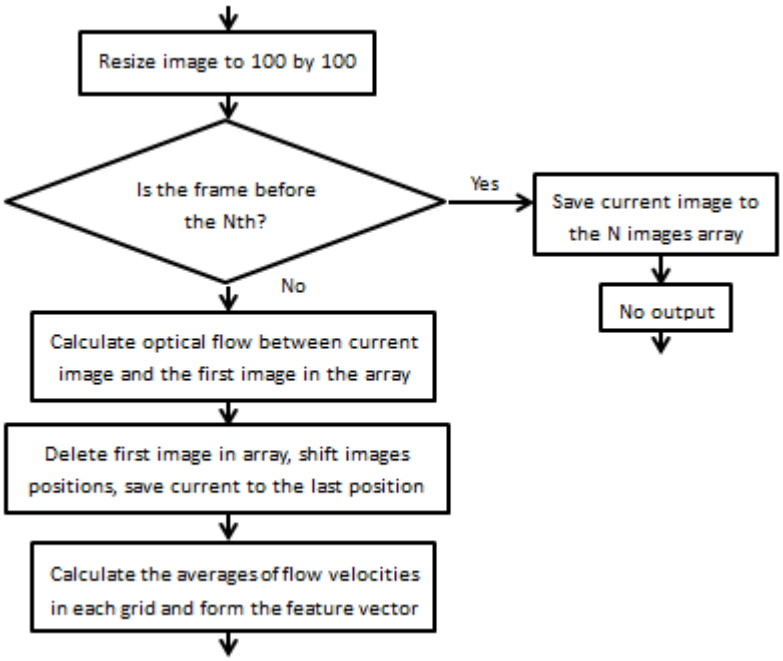


Figure 4.48: Flow chart inside Optical flow extraction box

4.7.3 Classification and Output

The feature vector of previous step is passing to the classifiers and the prediction will be given. One set of classifiers include six binary classifiers for six universal expressions. The feature vector will be feed into every classifier at the same time and every classifier gives a prediction of that expression. The results of one classifier will stack. Only more than one of past ten results are positive the expression will be shown on the expression indicator panel. The results of six classifiers are separate and will rank by the number of positive results and list in the panel. Two sets of classifiers can be used: CGP classifiers and SVM classifiers. They can be switched freely and active both or none of them.

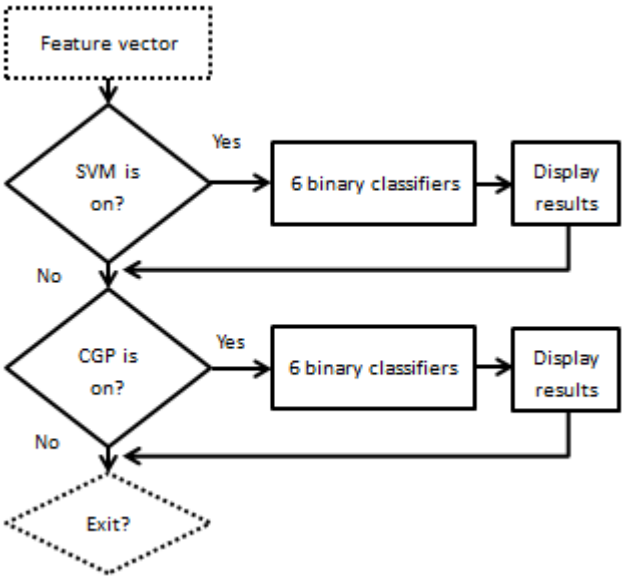


Figure 4.49: Flow chart inside classification box

The CGP classifier is AUC based, so no exact threshold is suggested by classifier. Instead the threshold can be selected freely by user. The strategy depends on the requirement of real applications. In this problem, a low pass classifier is better. The point selected on ROC is around 95% of TNR and 30% to 70% of TPR.

The GUI of the system is shown in figure 4.46 and 4.50. Press ‘F’ key to switch face detection on and off. The figure 4.46 is the non-face-detection mode; user should move the head or camera to make the face show within the blue square. Figure 4.51 is the system with face detection; this mode allows one user face appearing in the image and tracks the face. The face tracking is actually detecting face every frame which leads an unstable face position and size within the scan window. To address this, the old window size and position is stored and updated only when the new face scan window has more than 10% differences. The default scale of the video input is 2, so the system is processing the video stream at the resolution of 320 by 240 pixels. Press ‘B’ to active two classifier windows thus enter the expression detection mode. Pressing ‘S’ key will active the window of SVM classifier and close the window of CGP. ‘C’ key is for acting CGP window and shutting down SVM window. ‘ESC’ is the key to exit the programme. The system runs on the laptop purchased in 2009 with Intel Core 2 Duo 2.27 GHz CPU, 3.00 GB RAM and the Window 7-32bits. The FPS is around 17 without face detection and around 13 when face detection is on.

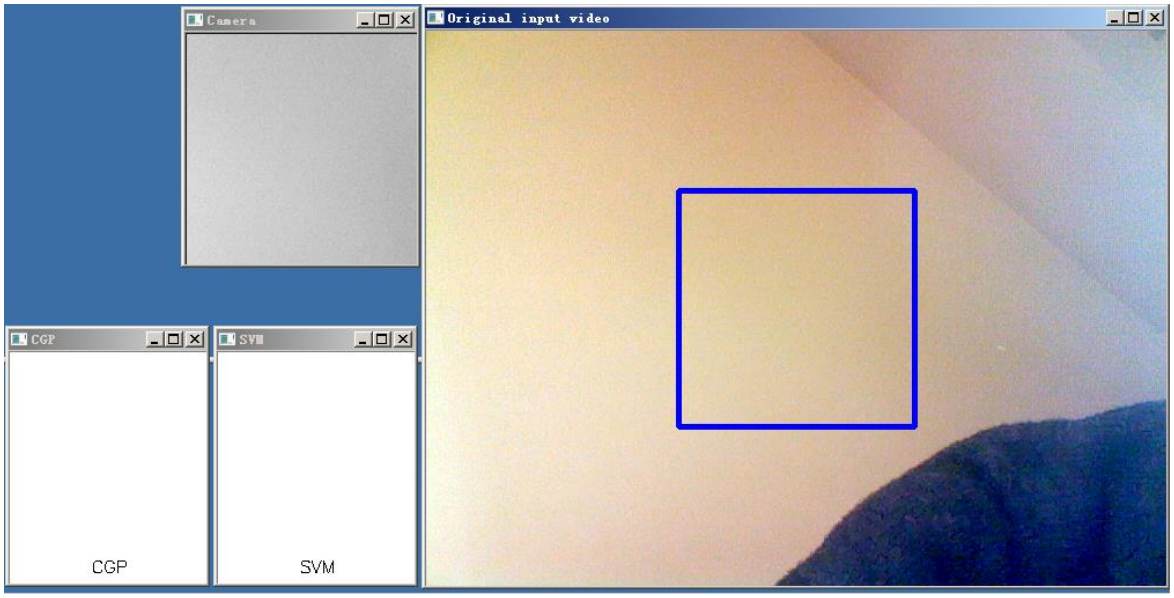


Figure 4.50: Expression recognition system (face detection is turned off)

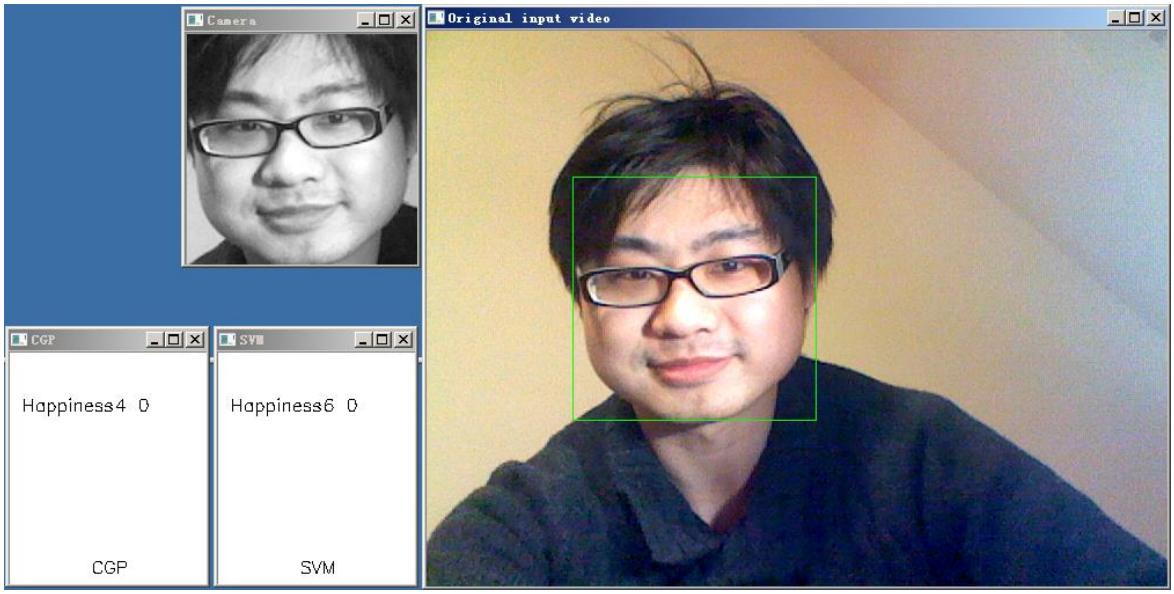


Figure 4.51: Expression recognition system with face detection

Another phenomenon is the offset of expression will be recognised as another expression. For example a face from neutral to smile will be recognised as happiness and from smile back to neutral will be misclassified as sadness. Then the flows of offset of expression are added in training data as the negative samples. Classifiers are trained by these data again and the new results reduce the phenomenon greatly.

4.8 Summary

In this chapter, Cartesian Genetic Programming (CGP) is introduced and proposed as a suitable structure for a classifier. The feature vector obtained from the optical flow, as described in chapter 3, is used as the input to CGP, and the output is converted to a predicted class label. In the training phase, the feature vector of the sample and its class label are already known. The count of correct classifications can be used to form the fitness function. The evolution of CGP is designed to find the solution that makes the most correct classifications on the training data. The test phase will use the network relating to the best solution obtained in the training phase, and predicts the class label from new input feature vectors.

This classifier is used to solve a series of segmentation problems. The figures show that the better the fitness of the training phase, the higher the accuracy on testing data. Therefore, the strategy of evolution that finds the solution with the best fitness is able to evolve a good classifier and give good classification results. But compared with SVM, it could not achieve better results, because SVM is naturally strong at this kind of problem. During these experiments, different combinations of CGP parameters have been tested. The parameters are also used for the following experiments.

A CGP classifier is then applied to expression recognition using optical flow based features. However, the imbalanced data set results in poor performance of the minority class accuracy. This is because the fitness function is guided by the number of correct classifications and does not take account of the accuracy of a minority class. Consequently, the AUC is considered as a more suitable fitness function. The results of AUC based CGP classifier are much better than that simply counting correct classifications on the imbalanced expression recognition problem.

The terminal condition of evolution process is also discussed. Longer evolution may make the classifier suffer from overfitting. Stopping evolution early may prevent overfitting to a certain extent. Box plots of 10 runs of results show that the median of 10,000 generations training is better than that of 1 million generations in 66% of the cases.

The results of the CGP classifier are then compared with the SVM classifier. The results show the CGP classifier can perform better than SVM because of the flexibility of CGP - it can apply a suitable fitness function for different situations with ease, which SVM cannot. Also, by adjusting the parameters of CGP can also lead to better performance in this problem.

The final accuracy is still not high enough and is probably due to the differences between two databases. An imbalanced training dataset and also the way in which sample data was obtained for this problem are probably relevant factors. The rising part of an expression is suitable for extracting optical flow and this flow information may be better than still image. However, because

of the different appearances of people and actions they make for an expression result in the static image analyse being unstable. The movement trend of expression from different people should be similar because the muscles that initiate a genuine expression are not controlled consciously. However, this method also causes some problems. It takes the optical flow between frames and each flow data is taken as one sample with one rising expression containing 10-20 frames. Only correctly recognised expressions in every frame over this period will result in a high recognition rate. As the beginning of an expression is very weak and hard to distinguish, if it is included in the recognition period, then it is understandable that the resulting recognition rate is not high.

An expression recognition system has been introduced and implemented based on the experiments. It can work automatically with the face detection function on. If the face detection function does not perform well, e.g. poor illumination condition or multiple faces presented, the system can be switch to non-face-detection mode (not automatic). Then the analysis will only focus on the centre area of the input stream, no matter if faces are there. This requires the user move the head or camera to make sure face is within the blue square in the centre. The recognition of expressions is from both SVM and CGP classifiers. 6 binary classifiers are used for detecting the appearance of corresponding expression, 12 classifiers if both SVM and CGP are active. The predictions are showed as the stack of previous 10 frames for each classifier and the results, counting of the positives of each classifier, are showed in a descending order.

5 Conclusion

Expression Recognition

Expression recognition is an important topic in image processing. The main techniques have been reviewed and introduced in the background chapter and it was noted that dynamic analysis has become more and more popular, in place of static image analysis. The dynamic techniques are not only focused on the appearances of the final state of expressions but also the transition between neutral and formed expressions. These methods achieved better performance because they reduced the variation in appearance of expressions between different people. The transition of expression should be similar because the genuine expression is generated by unconscious muscle actions. Therefore, dynamic analysis using the rising part of an expression should be more reliable.

Databases

The experiments used two expression video data sources: the MMI database and the FG-NET database. They provide a frontal face camera angle, a stable and fine illumination condition and good video quality. They both have a sufficient number of male and female candidates, but the MMI database has better diversity in the age and ethnicity of the candidates. They all provide at least six universal expressions (happiness, anger, surprise, fear, sadness and disgust [by Paul Ekman]). The FG-NET database attempts to record natural expressions instead of acted expressions. The MMI database initially consists of candidates' deliberate expressions but is supplemented by more natural expression videos.

A custom database was also created. The strategy adopted was to record video of the candidate's face while playing a series of short video clips to evoke a desired emotion. The candidate's facial transition was then recorded for use in subsequent expression research. In total 15 short clips were used for the whole process, including three clips for *happy*, two for *scary* and five other expressions. There is a five second rest between each clip to let the candidates recover to a normal emotion from the previous evoke expression. At the end of the video, candidates were asked to act the six expressions intentionally and were provided with an image of the corresponding expression in case they did not know how to perform the expression. The whole procedure is limited to within 15 minutes to avoid making people tired. In total 37 expression videos of the whole procedure from different candidates have been recorded, mostly university students in China. Although the video data is suitable for use in expression analysis, including the comparison between a real emotion and acting the expression, it has not been used in the

experimentation described in this thesis because the expressions recorded are too weak. It was, however, invaluable for understanding the limitations of evoking natural expressions compared to acted expressions. The existing databases tried several methods to make the expression performance better. However to generate strong and natural expression video is still a proble. The best way possibly is to create brand new and high quality evoking videos with psychologists' help; make sure the candidates fully cooperate and in a relax condition; the recording time should be as short as possible, this also require the evoking video can make people quick response. However the experience on how to create an expression video database has been obtained, it also helps to understand and have good use of other databases. The video data is weak but natural and it can be used for other suitable experiments in the future.

Analysis of Feature Extraction

The use of the Active Appearance Model (AAM) using sparse optical flow tracking on facial feature points is a popular technique; the feature points usually contain a mouth contour, eye corners and eye brows. However, it has limitations - the error of AAM matching feature points due to differences in people's appearance means that tracking only those feature points may not be sufficient. To address this issue, dense optical flow has been used for the experiments described in this thesis. Compared with optical flow, it is not necessary to know the exact feature points and flows of those feature points, which may contain useful information and can be caught. However, dense optical flow has its drawback - it contains too much data to be processed by the classifier, so an additional process is required to reduce it to a manageable size. Several methods have been investigated in this thesis including *clustering flows*, *identifying the five highest regions of flow* and the *average grid flow*. The *average grid flow* method worked better than the others, the main reason being that this feature selection method does not lose essential information, such as the location of the flows, and has an ordered feature vector which is fixed and therefore has clear meaning.

An evaluation of these data reduction methods was made by using them to train and test an SVM classifier as described in Chapter 3. The experiments consisted of six binary classifications for each expression, meaning that the two classes in the dataset are imbalanced, by approximately a 1:5 ratio. The data are the optical flow between frames in the rising period of expression. Each estimated dense optical flow between two frames is one sample in the dataset. The training of SVM with a RBF kernel is firstly achieved using 10-fold cross validation on the training data for determining the C and Gamma parameters. These parameters are then used for final training of the SVM using all training data. The *average grid flow* method achieves over 80% true positive rates and over 90% true negative rates on cross validation using the same database. As this

method is able to achieve high rates of expression recognition, it was used for subsequent experiments to compare CGP and SVM classifier performance.

Analysis of Classification

CGP is a highly efficient and flexible form of Genetic Programming, which has the ability to automatically evolve programs and was implemented as a classifier to investigate the hypothesis of this thesis. The inputs are feature vectors obtained from the pre-processing of the dense optical flow and the output is thresholded to represent one of two classes (although this is just one way of transforming the output value to a particular class). The training phase is the process of connecting CGP nodes through evolution to map input samples to their respective class labels. A fitness function evaluates the performance of the evolved solution after each generation until the best individual is obtained after the evolution process is completed. This best solution is then tested using unseen data, giving a class prediction.

The fitness function should be simple calculation, with clear meaning and aim, that allows improved in small steps during evolution. The fitness function first considered the number of correct recognitions in the training data. In 10 different runs of evolution in the segmentation experiments of Chapter 4, a higher fitness of counting correct classifications in the training has a higher accuracy on testing data. So the strategy of evolution that finds the solution with the best fitness is able to evolve good classifiers and give good classification results.

However, it did not perform well on the expression recognition which is a highly imbalanced data classification problem. Consequently, the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve was used as the fitness function. This technique considers the recognition rates of both classes and aims to have the best overall ROC after multiple thresholding. After training, the classifier can be set at any threshold to achieve the corresponding TPR and TNR. The results show that the accuracy and ROC curve of AUC-based CGP classifier are much better than counting correct responses on the imbalanced expression recognition problem.

Overfitting is the phenomenon where a classifier is mislead by the noise and error of data that does not describe the underlying relationship. This leads to poor predictive performance. To avoid overfitting, one possible method is stop the evolution process early. Two methods have been investigated in this thesis: reducing the number of generations used for evolution and increasing the final fitness as the terminal condition. The box plots for AUC test data over 10 runs showed that the median of the CGP classifier, trained after 10,000 generations, is higher than that achieved after 1 million generations in 66% of the time.

These show the flexibilities of CGP; it can be set different fitness functions to fit the specific problem, which conventional classifiers are not able to do. With these optimisations and adjustments, the CGP classifier is able to have better performance than SVM in this expression recognition task. From the results in Chapter 4, it can be seen that SVM has inferior performance compared to CGP, typically having a lower AUC. The CGP classifier train over 10,000 generations was found to have optimum performance. All CGP classifiers, trained over 1 million generations, 10,000 generations and terminated with a fitness of 0.1, have higher AUC than SVM in 64% to 75% of cases. However, in this application, left part of the ROC curve is more important than the top of the ROC curve, i.e., the area of the curve has a different weighting, meaning that the AUC cannot always represent the performance in this specific problem. The comparative performance of SVM has in two cases been underestimated due to this issue, but does not adversely affect the assessment of performance overall.

The results of training and testing using different databases is still disappointing. There are three reasons why such classification is difficult: the imbalanced nature of the datasets, inherent differences between two databases, and the way the training data is generated. The samples are taken from the optical flow between two frames in the transition of expressions. Each sample is one set of dense optical flow between two frames and one expression video clip contains many such samples. Only if almost all these samples are correctly recognised, can high classification rates be achieved. Obviously, this is not easy, especially at the very beginning of the transition as, at this time, it does not have enough discriminate information.

An Automatic Expression Recognition System

An expression recognition system has been implemented. The system takes a video stream from a camera or video file and tracks the face area and extracts features by the method described in Chapter 2. The expression will be estimated and shown in another window. The results are presented for six binary classifiers, each trained to recognise the six expressions separately. The type of classifier can be selected and switched during the video stream. If both types of classifier are active (CGP and SVM), there will be 12 binary classifiers working at the same time. The recognised expression will be reported if more than 2 frames of past 10 are indicated from the corresponding binary classifier. The AUC method gives a series of TPR and TNR pairs. Alternatively, a fixed threshold can be applied that provides the option to users to choose the settings most suitable one for a particular situation. The imbalanced datasets used in the training of the classifiers means that the minority class, also the expression to be detected, should be subject to a low pass rate to make the operation of the system more reliable.

Contribution

The work introduced a dense optical flow based dynamic expression recognition method. Compared with the popular sparse optical flow methods at the moment, it may require more processing time due to the higher computation, but it is able to extract flow information which may be important for recognition and is difficult for sparse optical flow to track.

CGP has been applied on expression recognition for the first time. The flexibility makes it easy to adjust and tuned for a better performance on specific problem. In the thesis, CGP classifier reduced the influences of imbalanced data by adjusting the fitness function to an AUC based one. The overfitting phenomenon was also reduced by tuning the parameters of CGP. Then the CGP based classifier can achieve better performance than SVM on the expression recognition experiments in this thesis. An automatic expression recognition system has also been built based on previous experiments. It runs 17/13 FPS without/with face detection function on a laptop, which has Intel Core 2 2.27GHz CPU.

Future Work

Ensemble Learning

The CGP classifier is flexible and has been optimised for this particular application. This is a distinct advantage of CGP over other conventional classifiers, such as SVM. Not only can the fitness function be changed for optimising the results, but also other parameters that determine the structure and operation of the network. The training strategy can be easily adapted to the specific problem. The result of this project has shown how an optimised CGP classifier can outperform SVM in more than one case, but the SVM classifier is also stronger in several experiments due to its own special characteristics. Since no single classifier has the best performance at all times, an ensemble classifier[146] could be used to combine the output from both classifier methods in order to achieve better overall results. Although the ensemble learning will increase computational effort, it will have the both weak classifiers' prediction, and then combine these to achieving a better performance.

Hypothesis Revisited

The hypothesis proposed in chapter 1 can now be reconsidered in the light of the experimental investigations presented in this thesis:

"Evolutionary Algorithms are an effective means of recognising and classifying human facial expression."

It is asserted that the work presented in this thesis does support the hypothesis and that the use of evolutionary algorithms such as CGP in the recognition of human facial expressions should be investigated further.

6 Definitions

ASM : Active Shape Model

AAM: Active Appearance Model

PDM: Point Distribution Model

FACS: Facial Action Coding System

FPS: Frames per Second

LK: Lucas – Kanade

LBP: Local Binary Pattern

SVM: Support Vector Machine

RBF: Radial Basis Function

NN: Neural Network

EAs: Evolutionary Algorithms

GP: Genetic Programming

CGP: Cartesian Genetic Programming

EP: Evolutionary Programming

ES: Evolutionary Strategies

GAs: Genetic Algorithms

AI: Artificial Intelligent

ROC: Recursive Operating Characteristic

AUC: Area under Curve

CGP-CC: CGP classifier with fitness function of counting correct classifications

CGP-AUC: CGP classifier with AUC as the fitness function

CGP-1M: CGP classifier trained with the terminal condition of reaching 1 million generations

CGP-10K: CGP classifier trained with the terminal condition of reaching 10,000 generations

CGP-F0.1: CGP classifier trained with the terminal condition of fitness reaching 0.1 (AUC value)

7 References

1. Picard, R.W., *Affective computing*. 2000: MIT press.
2. Scherer, K.R., *What are emotions? And how can they be measured?* Social Science Information, 2005. **44**: p. 693-727.
3. Darwin, C., *The Expression of the Emotions in Man and Animals*. 1872, London: John Murray.
4. Wikipedia. *Emotion*. 2014; Available from: http://en.wikipedia.org/wiki/Emotion.
5. Busso, C., et al. *Analysis of emotion recognition using facial expressions, speech and multimodal information*. in *Proceedings of the 6th international conference on Multimodal interfaces*. 2004. ACM.
6. Friesen, W.V., *Cultural differences in facial expressions in a social situation: An experimental test of the concept of display rules*. 1972, University of California: San Francisco.
7. Ekman, P., *Emotions Revealed*. 2003, New York: Henry Holt and Co. p. 237.
8. Pantic, M. and L.J.M. Rothkrantz, *Facial action recognition for facial expression analysis from static face images.* Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 2004. **34**(3): p. 1449-1461.
9. Luo, R.C., C.Y. Huang, and P.H. Lin. *Alignment and tracking of facial features with component-based active appearance models and optical flow*. in *Advanced Intelligent Mechatronics (AIM), 2011 IEEE/ASME International Conference on*. 2011.
10. Dehkordi, B.K. and J. Haddadnia. *Facial expression recognition in video sequence images by using optical flow*. in *Signal Processing Systems (ICSPS), 2010 2nd International Conference on*. 2010.
11. Lee, B., J. Chun, and P. Park. *Classification of Facial Expression Using SVM for Emotion Care Service System*. in *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. SNPD '08. Ninth ACIS International Conference on*. 2008.
12. Shan, C., S. Gong, and P.W. McOwan. *Robust facial expression recognition using local binary patterns*. in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*. 2005.
13. Zhao, G. and M. Pietikainen, *Dynamic Texture Recognition Using Local Binary Patterns with an Application to Facial Expressions.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2007. **29**(6): p. 915-928.
14. Ruo, D., et al. *Facial expression recognition using histogram variances faces*. in *Applications of Computer Vision (WACV), 2009 Workshop on*. 2009.
15. Edwards, G.J., C.J. Taylor, and T.F. Cootes. *Interpreting face images using active appearance models*. in *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*. 1998.
16. Yongmian, Z. and J. Qiang, *Active and dynamic information fusion for facial expression understanding from image sequences.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2005. **27**(5): p. 699-714.
17. Shan, C., S. Gong, and P.W. McOwan, *Dynamic facial expression recognition using a Bayesian temporal manifold model.* Proc. Conf. Brit., 2006: p. 297-306.
18. Yan, T., L. Wenhui, and J. Qiang, *Facial Action Unit Recognition by Exploiting Their Dynamic and Semantic Relationships.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2007. **29**(10): p. 1683-1699.
19. Neggaz, N., M. Besnassi, and A. Benyettou, *Application of improved AAM and probabilistic neural network to facial expression recognition.* Applied Sciences, 2010. **10**: p. 1572-1579.

20. Ma, L. and K. Khorasani, *Facial expression recognition using constructive feedforward neural networks.* Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 2004. **34**(3): p. 1588-1595.
21. Suzuki, K., H. Yamada, and S. Hashimoto, *A similarity-based neural network for facial expression analysis.* Pattern Recognition Letters, 2007. **28**(9): p. 1104-1111.
22. Fragopanagos, N. and J.G. Taylor, *Emotion recognition in human–computer interaction.* Neural Networks, 2005. **18**(4): p. 389-405.
23. Eiben, A.E. and J.E. Smith, *Introduction to Evolutionary Computing*. 2003: Springer, Natural Computing Series.
24. Alba, E., *Parallel evolutionary algorithms can achieve super-linear performance.* Information Processing Letters, 2002. **82**(1): p. 7-13.
25. Tsymbal, A., M. Pechenizkiy, and P. Cunningham, *Diversity in search strategies for ensemble feature selection.* Information fusion, 2005. **6**(1): p. 83-98.
26. Viola, P. and M.J. Jones, *Robust Real-Time Face Detection.* International Journal of Computer Vision, 2004. **57**(2): p. 137-154.
27. Ephraim, T., T. Himmelman, and K. Siddiqi. *Real-Time Viola-Jones Face Detection in a Web Browser*. in *Computer and Robot Vision, 2009. CRV '09. Canadian Conference on*. 2009.
28. Liying, L. and G. Weiwei. *Study of Face Detection Algorithm for Real-time Face Detection System*. in *Electronic Commerce and Security, 2009. ISECS '09. Second International Symposium on*. 2009.
29. Ruian, L., Z. Mimi, and M. Shengtao. *Design of face detection and tracking system*. in *Image and Signal Processing (CISP), 2010 3rd International Congress on*. 2010.
30. Bilaniuk, O., et al. *Fast LBP Face Detection on Low-Power SIMD Architectures*. in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*. 2014.
31. Ekman, P. and W.V. Friesen, *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. 1978: Consulting Psychologists Press.
32. Ryan, A., et al. *Automated Facial Expression Recognition System*. in *Security Technology, 2009. 43rd Annual 2009 International Carnahan Conference on*. 2009.
33. Donato, G., et al., *Classifying facial actions.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1999. **21**(10): p. 974-989.
34. Pantic, M. and I. Patras. *Detecting facial actions and their temporal segments in nearly frontal-view face image sequences*. in *Systems, Man and Cybernetics, 2005 IEEE International Conference on*. 2005.
35. Robinson, J.A., *Covariance matrix estimation for appearance-based face image processing*, in *Proc.British Machine Vision Conference*. 2005. p. 389-398.
36. Robinson, J.A. *Regularized single-kernel conditional density estimation for face description*. in *Image Processing (ICIP), 2009 16th IEEE International Conference on*. 2009.
37. Robinson, J.A. and J.R. Hyde, *Estimation of face depths by conditional densities*, in *British Machine Vision Conference*. 2005.
38. Murthy, G.R.S. and R.S. Jadon, *Effectiveness of eigenspaces for facial expressions recognition.* International Journal of Computer Theory and Engineering, 2009. **1**: p. 638-642.
39. Pantic, M. and L.J.M. Rothkrantz, *Toward an affect-sensitive multimodal human-computer interaction.* Proceedings of the IEEE, 2003. **91**(9): p. 1370-1390.
40. Castellano, G., L. Kessous, and G. Caridakis, *Emotion Recognition through Multiple Modalities: Face, Body Gesture, Speech Affect and Emotion in Human-Computer Interaction*, C. Peter and R. Beale, Editors. 2008, Springer Berlin / Heidelberg. p. 92-103.
41. Caridakis, G., K. Karpouzis, and S. Kollias, *User and context adaptive neural networks for emotion recognition.* Neurocomputing, 2008. **71**(13–15): p. 2553-2562.
42. Pighin, F., R. Szeliski, and D.H. Salesin, *Modeling and Animating Realistic Faces from Images.* International Journal of Computer Vision, 2002. **50**(2): p. 143-169.
43. Tao, H. and T.S. Huang, *Visual Estimation and Compression of Facial Motion Parameters—Elements of a 3D Model-Based Video Coding System.* International Journal of Computer Vision, 2002. **50**(2): p. 111-125.
44. Cohen, I., et al., *Facial expression recognition from video sequences: temporal and static modeling.* Computer Vision and Image Understanding, 2003. **91**(1–2): p. 160-187.

45. Dornaika, F. and B. Raducanu, *Inferring facial expressions from videos: Tool and application.* Signal Processing: Image Communication, 2007. **22**(9): p. 769-784.
46. Sun, Y. and L. Yin, *Facial Expression Recognition Based on 3D Dynamic Range Model Sequences*, in *Proceedings of the 10th European Conference on Computer Vision: Part II*. 2008, Springer-Verlag: Marseille, France. p. 58-71.
47. Ya, C., H. Changbo, and M. Turk. *Probabilistic expression analysis on manifolds*. in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. 2004.
48. Raducanu, B. and F. Dornaika. *Dynamic facial expression recognition using Laplacian Eigenmaps-based manifold learning*. in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. 2010.
49. Fasel, I.R., M.S. Bartlett, and J.R. Movellan. *A comparison of Gabor filter methods for automatic detection of facial landmarks*. in *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*. 2002.
50. Oshidari, B. and B.N. Araabi. *An effective feature extraction method for facial expression recognition using adaptive Gabor wavelet*. in *Progress in Informatics and Computing (PIC), 2010 IEEE International Conference on*. 2010.
51. Wu, P., et al. *Face Expression Recognition Based on Feature Fusion*. in *Intelligent Systems and Applications, 2009. ISA 2009. International Workshop on*. 2009.
52. Yu, J. and B. Bhanu, *Evolutionary feature synthesis for facial expression recognition.* Pattern Recognition Letters, 2006. **27**(11): p. 1289-1298.
53. Shan, C., S. Gong, and P.W. McOwan, *Facial expression recognition based on Local Binary Patterns: A comprehensive study.* Image and Vision Computing, 2009. **27**(6): p. 803-816.
54. Hyun-Chul, C. and O. Se-Young. *Facial Identity and Expression Recognition by using Active Appearance Model with Efficient Second Order Minimization and Neural Networks*. in *Computational Intelligence in Robotics and Automation, 2007. CIRA 2007. International Symposium on*. 2007.
55. Zalewski, L. and S. Gong. *Synthesis and recognition of facial expressions in virtual 3D views*. in *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*. 2004.
56. He, K., G. Wang, and Y. Yang. *Optical flow-based facial feature tracking using prior measurement*. in *Cognitive Informatics, 2008. ICCI 2008. 7th IEEE International Conference on*. 2008.
57. Ojala, T., M. Pietikainen, and D. Harwood. *Performance evaluation of texture measures with classification based on Kullback discrimination of distributions*. in *Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*. 1994.
58. Cootes, T., et al., *Active Shape Models - Their Training and Application.* Computer Vision and Image Understanding, 1995. **61**: p. 18-23.
59. Cootes, T., G. Edwards, and C. Taylor, *Active appearance models Computer Vision — ECCV'98*, H. Burkhardt and B. Neumann, Editors. 1998, Springer Berlin / Heidelberg. p. 484-498.
60. Sung, J., S. Lee, and D. Kim. *A Real-Time Facial Expression Recognition using the STAAM*. in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*. 2006.
61. Robin, T., M. Bierlaire, and J. Cruz, *Facial expression recognition with a discrete choice model.* Journal of Choice Modelling, 2010. **2**(1): p. 95-148.
62. Horn, B.K.P. and B.G. Schunck, *Determining optical flow.* Artificial Intelligence, 1981. **17**: p. pp 185-203.
63. Lucas, B.D. and T. Kanade, *An iterative image registration technique with an application to stereo vision.* Proceedings of Imaging Understanding Workshop, 1981: p. pp. 121-130.
64. Peng, X.N., et al., *Research on dynamic facial expressions recognition* Modern Applied Science, 2009. **3**(5): p. 31-37.
65. Kong, J., Y.-z. Zhan, and Y.-b. Chen. *Expression Recognition Based on VLBP and Optical Flow Mixed Features*. in *Image and Graphics, 2009. ICIG '09. Fifth International Conference on*. 2009.

66. Wang, G., G. Yang, and K. Fu. *Facial Expression Recognition Based on Extended Optical Flow Constraint*. in *Intelligent Computation Technology and Automation (ICICTA), 2010 International Conference on*. 2010.
67. Anderson, K. and P.W. McOwan, *A real-time automated system for the recognition of human facial expressions.* Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 2006. **36**(1): p. 96-105.
68. Shreve, M., et al. *Towards macro- and micro-expression spotting in video using strain patterns*. in *Applications of Computer Vision (WACV), 2009 Workshop on*. 2009.
69. Hsieh, C.-K., S.-H. Lai, and Y.-C. Chen. *Expressional face image analysis with constrained optical flow*. in *Multimedia and Expo, 2008 IEEE International Conference on*. 2008.
70. Scholkopf, B., et al., *Comparing support vector machines with Gaussian kernels to radial basis function classifiers.* Signal Processing, IEEE Transactions on, 1997. **45**(11): p. 2758-2765.
71. Chen, P., et al., *A tutorial on ν-support vector machines.* Applied Stochastic Models in Business and Industry, 2005: p. 111-136.
72. Darwin, C., *On The Origin of Species*. Vol. XIV. 1859.
73. Hirsh, H., *Trends and Controversies: Genetic Programming.* IEEE Intelligent Systems, 2000. **15**(3): p. 74-84.
74. Weimer, W., et al. *Automatically finding patches using genetic programming*. in *Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on*. 2009.
75. Shichel, Y., E. Ziserman, and M. Sipper, *GP-Robocode: Using Genetic Programming to Evolve Robocode Players*, in *Genetic Programming*, M. Keijzer, et al., Editors. 2005, Springer Berlin Heidelberg. p. 143-154.
76. Tanev, I., *Incorporating Learning Probabilistic Context-Sensitive Grammar in Genetic Programming for Efficient Evolution and Adaptation of Snakebot*, in *Genetic Programming*, M. Keijzer, et al., Editors. 2005, Springer Berlin Heidelberg. p. 155-166.
77. Hauptman, A. and M. Sipper, *GP-EndChess: Using Genetic Programming to Evolve Chess Endgame Players*, in *Genetic Programming*, M. Keijzer, et al., Editors. 2005, Springer Berlin Heidelberg. p. 120-131.
78. Azaria, Y. and M. Sipper, *GP-Gammon: Using Genetic Programming to Evolve Backgammon Players*, in *Genetic Programming*, M. Keijzer, et al., Editors. 2005, Springer Berlin Heidelberg. p. 132-142.
79. Jackson, D., *Evolving Defence Strategies by Genetic Programming*, in *Genetic Programming*, M. Keijzer, et al., Editors. 2005, Springer Berlin Heidelberg. p. 281-290.
80. Poli, R., W. Langdon, and O. Holland, *Extending Particle Swarm Optimisation via Genetic Programming*, in *Genetic Programming*, M. Keijzer, et al., Editors. 2005, Springer Berlin Heidelberg. p. 291-300.
81. Winkeler, J.F. and B.S. Manjunath, *Genetic Programming for Object Detection*, in *Genetic Programming 1997: Proceedings of the Second Annual Conference*. 1997, Morgan Kaufmann. p. 330--335.
82. Sherrah, J.R., R.E. Bogner, and A. Bouzerdoum, *The Evolutionary Pre-Processor: Automatic Feature Extraction for Supervised Classification using Genetic Programming*, in *In Proc. 2nd International Conference on Genetic Programming*. 1997, Morgan Kaufmann. p. 304--312.
83. Guo, L., et al., *Automatic feature extraction using genetic programming: An application to epileptic EEG classification.* Expert Systems with Applications, 2011. **38**(8): p. 10425-10436.
84. Akyol, A., Y. Yaslan, and O. Erol, *A Genetic Programming Classifier Design Approach for Cell Images*, in *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, K. Mellouli, Editor. 2007, Springer Berlin Heidelberg. p. 878-888.
85. Hong, G. and A.K. Nandi. *Breast Cancer Diagnosis Using Genetic Programming Generated Feature*. in *Machine Learning for Signal Processing, 2005 IEEE Workshop on*. 2005.
86. Muni, D.P., N.R. Pal, and J. Das, *Genetic programming for simultaneous feature selection and classifier design.* Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 2006. **36**(1): p. 106-117.

87. Lensberg, T. and Klaus Reiner Schenk-Hoppe", *Hedging without sweat: a genetic programming approach.* Quantitative Finance Letters, 2013. **1**: p. 41-46.
88. Zhang, L. and A.K. Nandi, *Fault classification using genetic programming.* Mechanical Systems and Signal Processing, 2007. **21**(3): p. 1273--1284.
89. Winkler, S.M., M. Affenzeller, and S. Wagner, *Using enhanced genetic programming techniques for evolving classifiers in the context of medical diagnosis.* Genetic Programming and Evolvable Machines, 2009. **10**(2): p. 111-140.
90. Nandi, R.J., et al., *Classification of breast masses in mammograms using genetic programming and feature selection.* Medical and Biological Engineering and Computing, 2006. **44**(8): p. 683-694.
91. Hong, J.-H. and S.-B. Cho, *The classification of cancer based on DNA microarray data that uses diverse ensemble genetic programming.* Artif. Intell. Med., 2006. **36**(1): p. 43-58.
92. Paul, T.K. and H. Iba, *Prediction of Cancer Class with Majority Voting Genetic Programming Classifier Using Gene Expression Data.* Computational Biology and Bioinformatics, IEEE/ACM Transactions on, 2009. **6**(2): p. 353-367.
93. Gathercole, C. and P. Ross, *Small Populations over Many Generations can beat Large Populations over Few Generations in Genetic Programming*, in *Genetic Programming 1997: Proceedings of the Second Annual Conference*, J.R. Koza, et al., Editors. 1997, Morgan Kaufmann.
94. Burke, E.K., S. Gustafson, and G. Kendall, *Diversity in genetic programming: an analysis of measures and correlation with fitness.* Evolutionary Computation, IEEE Transactions on, 2004. **8**(1): p. 47-62.
95. Brameier, M. and W. Banzhaf, *A comparison of linear genetic programming and neural networks in medical data mining.* Evolutionary Computation, IEEE Transactions on, 2001. **5**(1): p. 17-26.
96. Langdon, W.B. and R. Poli, *An analysis of the MAX problem in genetic programming*, in *Genetic Programming 1997: Proceedings of the Second Annual Conference*. 1997, Morgan Kaufmann. p. 220-230.
97. Luke, S. and L. Spector, *A Comparison of Crossover and Mutation in Genetic Programming*, in *Genetic Programming 1997: Proceedings of the Second Annual Conference*. 1997, Morgan Kaufmann. p. 240-248.
98. Punch, W.F., *How Effective are Multiple Populations in Genetic Programming*, in *Genetic Programming 1998: Proceedings of the Third Annual Conference*, J.R. Koza, et al., Editors. 1998, Morgan Kaufmann. p. 308-313.
99. Koza, J.R., *Genetic Programming*. 1992, Bradford: MIT Press.
100. Jabeen, H. and A.R. Baig, *Review of Classification Using Genetic Programming.* International Journal of Engineering Science and Technology, 2010. **2**(2): p. 94-103.
101. Bojarczuka, C.C., H.S. Lopesb, and A.A. Freitasc, *Data mining with constrained-syntax genetic programming: applications in medical data set.* algorithms, 2001. **6**(7).
102. Lin, J.-Y., et al., *Classifier design with feature selection and feature extraction using layered genetic programming.* Expert Systems with Applications, 2007.
103. Muni, D.P., N.R. Pal, and J. Das, *A novel approach to design classifiers using genetic programming.* Evolutionary Computation, IEEE Transactions on, 2004. **8**(2): p. 183-196.
104. Tsakonas, A., *A comparison of classification accuracy of four genetic programming-evolved intelligent structures.* Information Sciences, 2006. **176**(6): p. 691-724.
105. Wongseree, W., et al., *Thalassaemia classification by neural networks and genetic programming.* Inf. Sci., 2007. **177**(3): p. 771-786.
106. Berlanga, F.J., et al., *GP-COACH: Genetic Programming-based learning of COmpact and ACcurate fuzzy rule-based classification systems for High-dimensional problems.* Inf. Sci., 2010. **180**(8): p. 1183-1200.
107. Mendes, R.R.F., et al., *Discovering Fuzzy Classification Rules with Genetic Programming and Co-evolution*, in *Principles of Data Mining and Knowledge Discovery*, L. De Raedt and A. Siebes, Editors. 2001, Springer Berlin Heidelberg. p. 314-325.
108. Deb, K., et al., *A fast and elitist multiobjective genetic algorithm: NSGA-II.* Evolutionary Computation, IEEE Transactions on, 2002. **6**(2): p. 182-197.

109. Parrott, D., L. Xiaodong, and V. Ciesielski. *Multi-objective techniques in genetic programming for evolving classifiers*. in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*. 2005.
110. Zhang, M. and U. Bhowan, *Program Size and Pixel Statistics in Genetic Programming for Object Detection*, in *Applications of Evolutionary Computing*, G. Raidl, et al., Editors. 2004, Springer Berlin Heidelberg. p. 379-388.
111. Lichodzijewski, P. and M.I. Heywood, *Pareto-coevolutionary genetic programming for problem decomposition in multi-class classification*, in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. 2007, ACM: London, England. p. 464-471.
112. Loveard, T. and V. Ciesielski. *Representing classification problems in genetic programming*. in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*. 2001.
113. Chi, Z., et al., *Evolving accurate and compact classification rules with gene expression programming.* Evolutionary Computation, IEEE Transactions on, 2003. **7**(6): p. 519-531.
114. Zhang, M. and W. Smart, *Multiclass Object Classification Using Genetic Programming*, in *Applications of Evolutionary Computing*, G. Raidl, et al., Editors. 2004, Springer Berlin Heidelberg. p. 369-378.
115. Chawla, N.V., N. Japkowicz, and A. Kotcz, *Editorial: special issue on learning from imbalanced data sets.* SIGKDD Explor. Newsl., 2004. **6**(1): p. 1-6.
116. Patterson, G. and M. Zhang, *Fitness Functions in Genetic Programming for Classification with Unbalanced Data*, in *AI 2007: Advances in Artificial Intelligence*, M. Orgun and J. Thornton, Editors. 2007, Springer Berlin Heidelberg. p. 769-775.
117. Kishore, J.K., et al., *Application of genetic programming for multicategory pattern classification.* Evolutionary Computation, IEEE Transactions on, 2000. **4**(3): p. 242-258.
118. Hawkins, D.M., *The Problem of Overfitting.* Journal of Chemical Information and Modeling, 2004. **44**(1): p. 1-12.
119. Paris, G., D. Robilliard, and C. Fonlupt, *Exploring Overfitting in Genetic Programming*, in *Artificial Evolution*, P. Liardet, et al., Editors. 2004, Springer Berlin Heidelberg. p. 267-277.
120. Smith, M. and L. Bull, *Genetic Programming with a Genetic Algorithm for Feature Construction and Selection.* Genetic Programming and Evolvable Machines, 2005. **6**(3): p. 265-281.
121. Miller, J. and P. Thomson, *Cartesian Genetic Programming*, in *Genetic Programming*, R. Poli, et al., Editors. 2000, Springer Berlin Heidelberg. p. 121-132.
122. Salahuddin, S. and M.M. Khan. *Object categorization using Cartesian Genetic Programming (CGP) and CGP-Evolved Artificial Neural Network*. in *Computer Applications and Industrial Electronics (ICCAIE), 2010 International Conference on*. 2010.
123. Khan, M.M., G.M. Khan, and J.F. Miller. *Evolution of neural networks using Cartesian Genetic Programming*. in *Evolutionary Computation (CEC), 2010 IEEE Congress on*. 2010.
124. Harding, S. and J. Miller, *Evolution of Robot Controller Using Cartesian Genetic Programming*, in *Genetic Programming*, M. Keijzer, et al., Editors. 2005, Springer Berlin Heidelberg. p. 62-73.
125. Zangeneh, L. and P. Bentley, *Analyzing the Credit Default Swap Market Using Cartesian Genetic Programming*, in *Parallel Problem Solving from Nature, PPSN XI*, R. Schaefer, et al., Editors. 2010, Springer Berlin Heidelberg. p. 434-444.
126. Harding, S. *Evolution of image filters on graphics processor units using Cartesian Genetic Programming*. in *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*. 2008.
127. Harding, S. and W. Banzhaf, *Fast Genetic Programming on GPUs*, in *Genetic Programming*, M. Ebner, et al., Editors. 2007, Springer Berlin Heidelberg. p. 90-101.
128. Harding, S.L., J.F. Miller, and W. Banzhaf, *Self-modifying cartesian genetic programming*, in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. 2007, ACM: London, England. p. 1021-1028.
129. Walker, J.A., J.F. Miller, and R. Cavill, *A multi-chromosome approach to standard and embedded cartesian genetic programming*, in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. 2006, ACM: Seattle, Washington, USA. p. 903-910.

130. Lones, M.A., et al. *Discriminating normal and cancerous thyroid cell lines using implicit context representation Cartesian genetic programming*. in *Evolutionary Computation (CEC), 2010 IEEE Congress on*. 2010.
131. Ashmore, L. and J.F. Miller, *Evolutionary Art with Cartesian Genetic Programming*. 2004: Online technical report available from https://sites.google.com/site/julianfrancismiller/publications.
132. Ekman, P., *Universal Facial Expression in Emotion.* Studia Psychologica, 1973.
133. Pantic, M., et al. *Web-based database for facial expression analysis*. in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*. 2005.
134. Wallhoff, F., *Fgnet-facial expression and emotion database*. 2004, Technische Universität München.
135. Farneback, G., *Two-frame motion estimation based on polynomial expansion*, in *Proceedings of the 13th Scandinavian conference on Image analysis*. 2003, Springer-Verlag: Halmstad, Sweden. p. 363-370.
136. DevZone, O. *OpenCV Change Logs*. 2014; Available from: http://code.opencv.org/projects/opencv/wiki/ChangeLog#240.
137. Steinbach, M., G. Karypis, and V. Kumar. *A comparison of document clustering techniques*. in *KDD workshop on text mining*. 2000. Boston, MA.
138. Wikipedia. *Cluster analysis*. 2014; Available from: http://en.wikipedia.org/w/index.php?title=Cluster_analysis&oldid=631785337.
139. OpenCV. *OpenCV (Open Source Computer Vision)*. 2014; Available from: http://opencv.org/.
140. Wikipedia. *Receiver operating characteristic*. 2014; Available from: http://en.wikipedia.org/w/index.php?title=Receiver_operating_characteristic&oldid=629228225.
141. Hewlett, G., K.G. Stünkel, and H. Dieter Schlumberger, *A method for the quantitation of interleukin-2 activity.* Journal of Immunological Methods, 1989. **117**(2): p. 243-246.
142. Williamson, D.F., R.A. Parker, and J.S. Kendrick, *The box plot: a simple visual method to interpret data.* Annals of internal medicine, 1989. **110**(11): p. 916-921.
143. Wikipedia. *Box plot*. 2014; Available from: http://en.wikipedia.org/w/index.php?title=Box_plot&oldid=617605869.
144. Intel. *Intel C and C++ Compilers*. 2014; Available from: https://software.intel.com/en-us/c-compilers.
145. N8HPC. *N8 High Performance Computing*. 2014; Available from: http://n8hpc.org.uk/.
146. Shen, H.-B. and K.-C. Chou, *Ensemble classifier for protein fold pattern recognition.* Bioinformatics, 2006. **22**(14): p. 1717-1722.