

Optical Flow-based Facial Feature Tracking to Recognize AUs Modeled by Bayesian Networks

by

Xiaofan Sun

**A thesis submitted in partial satisfaction
of the requirements for the degree of**

Master of Science

Presented at

**Delft University of Technology,
Faculty of Electrical Engineering,
Mathematics, and Computer Science,
Man-Machine Interaction Group.**

June 2009



Man-Machine Interaction Group

Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology
Mekelweg 4
2628 CD Delft
The Netherlands

Philips Research

High Tech Campus 5
5656 AE Eindhoven
The Netherlands

Members of the Supervising Committee

Prof. drs. dr. L.J.M. Rothkrantz,
Dr.ir.P.Wiggers
Ir. H.Geers
Dr.ir.R. Braspenning
Ir. M.Popa

Copyright @ June 2009
Xiaofan Sun
1385593

Keywords: Action Unit (AU), Bayesian Network (BN), emotion recognition, Expert System (ES), facial expression, image processing, region of interest.

Abstract

Optical Flow-based Facial Feature Tracking to Recognize AUs Modeled by Bayesian Networks

Copyright @ 2009 by Xiaofan Sun (1385593)
Man-Machine Interaction Group
Faculty of EEMCS
Delft University of Technology

Members of the Supervising Committee

Prof. drs. dr. L.J.M. Rothkrantz,
Dr.ir.P.Wiggers
Ir. H.Geers
Dr.ir.R.Braspenning
Ir. M.Popa

This thesis describes a Bayesian Network (BN) model for recognizing the “Action Units (AUs)” of a facial expression using video sequence images as input. Features were extracted by using an optimal estimation optical flow method coupled with a physical (muscle) model describing the facial structure. The muscle action patterns are used for analysis, recognition, and synthesis of facial expressions. In the thesis the main approaches to facial expression recognition of dynamic images are designed considering three main parts: 1) Region of Interest Selection, 2) Feature Extraction, and 3) Image Classification.

Bayesian Networks are a powerful and flexible methodology for representing and computing with probabilistic models of a stochastic process. In the past decade, there has been increasing interest in applying them to practical problems, and this thesis shows that they can be used effectively in the field of automatic AU's recognition.

In past decade optical flows have been used to either model muscle activities or estimate the displacements of feature points but in this thesis we defined nine regions of interest (ROI) which contains the most complex motion by using entropy maximum algorithm. Furthermore, the results were statistically analyzed by compass diagrams to find out the major ranges of directions and velocities of vector flows in each ROI. We found that for the six basic emotions, the ROI are different, so we did not consider all of nine regions for every emotion because of the complexity of our model.

Furthermore, we present a methodology for obtaining the BN structure, learning the parameters and inference, including issues such as the discretization of continuous variables. Finally, we apply the BN model to recognize single Action Units (AUs) and some important AU combinations. The average classification rate for the single AUs is between 80% and 90% and for the AU combinations is above 90%.

TABLE OF CONTENTS

ABSTRACT	5
ACKNOWLEDGEMENT	11
CHAPTER 1	13
INTRODUCTION	13
1.1 PROBLEM DESCRIPTION	14
1.2 PROPOSED SOLUTION.....	15
1.3 THESIS MOTIVATION.....	15
1.4 THESIS OVERVIEW	16
1.5 THESIS CONTRIBUTIONS	17
CHAPTER 2	19
OVERVIEW OF RELATED WORK	19
2.1 FACIAL FEATURE MEASUREMENT	19
2.2 CLASSIFICATION	20
2.3 FACIAL EXPRESSION RECOGNITION.....	21
2.4 FUSION IN EMOTION MULTIMODAL RECOGNITION	22
2.4.1 Feature-Level Fusion	22
2.4.2 Decision-Level Fusion	23
2.4.3 Model-Level Fusion	24
CHAPTER 3	25
THEORETICAL BACKGROUND & METHOD OVERVIEW	25
3.1 LUCAS-KANADE METHOD	25
3.2 BAYESIAN NETWORK.....	26
3.3 DYNAMIC BAYESIAN NETWORK	27
3.4 PROBABILISTIC REASONING.....	28
3.4.1 Constructing a Bayesian Network.....	28
3.4.2 Steps to Follow in Constructing a Bayesian Network.....	29
3.4.3 Filling in Conditional Probability Tables.....	30
3.4.4 Exact Inference in Bayesian Networks	30
3.4.5 Bayesian Network Learning.....	31
3.4.6 Learning Problem.....	31
3.4.6.1 Structure Learning	31
3.4.6.2 Parameter Learning	32
3.4.6.3 Expectation Maximization (EM algorithm) Overview.....	32

3.4.7 Calculating Log Likelihood	33
3.4.8 CPTs and the Joint Probability Distribution.....	33
3.5 EXPERT SYSTEMS	35
3.5.1 Environment.....	35
3.6 EXPERT SYSTEM BY CLIPS	36
3.6.1 Inference Engine	36
3.6.2 Knowledge Base	36
3.6.3 Features List.....	36
CHAPTER 4.....	37
RESEARCH GOAL AND ARCHITECTURE	37
4.1 RESEARCH ARCHITECTURE.....	37
4.2 RESEARCH GOAL	39
4.3 RESEARCH OVERVIEW	40
CHAPTER 5.....	43
REGION OF INTEREST AND FEATURE EXTRACTION.....	43
5.1 OVERVIEW	43
5.2 FACIAL LANDMARKS LOCALIZATION	44
5.2.1 Image Preprocessing	44
5.2.1.1 Image Alignment.....	45
5.2.1.2 Computing an Average Image.....	46
5.2.2 Algorithm of Facial Landmarks Localization	46
5.3 REGION OF INTEREST SELECTION AND LOCALIZATION.....	48
5.3.1 Optical Flow Regions in Face Mask	48
5.3.2 Divided Regions in Face	49
5.3.3 Most Complex Motion Region (ROI) Selection	50
5.3.3.1 Entropy Maximum Algorithm.....	50
5.3.3.2 Inter-Frame Method.....	51
5.3.4 Automatic Localization of ROI from AU-coded Facial Images	52
5.4 FEATURE EXTRACTION	54
5.5 OPTICAL FLOWS ANALYSIS	55
5.5.1 Optical Flow Detection and Presentation.....	55
5.5.2 Vector Flow Map and Analysis.....	56
5.5.3 Vector Flow Analysis in Region of Interest.....	59
CHAPTER 6.....	67
FACIAL EXPRESSION MODEL AND IMPLEMENTATION IN SMILE&GENIE	67
6.1 OVERVIEW	67
6.2 THE MULTI-LAYERED APPROACH	68
6.3 MODULE DESIGN	68
6.4 BAYESIAN NETWORK MODELING.....	69

6.5 DYNAMIC BAYESIAN NETWORK MODELING	72
6.6 MODEL SETUP IN GENIE & SMILE	73
6.7 SOFTWARE DESIGN	75
6.7.1 Parameters Estimation.....	77
CHAPTER 7.....	79
EXPERIMENTAL RESULTS.....	79
7.1 COHN-KANADE DATASET	79
7.2 LOCALIZATION OF FACIAL LANDMARKS FOR AUs RECOGNITION	80
7.2.1 Performance Evaluation Measure	80
7.2.2 Evaluation Results and Conclusion.....	81
7.3 AUs AND AU COMBINATIONS RECOGNITION	83
7.3.1 Training Dataset Design.....	83
7.3.2 Features Weighted.....	83
7.3.3 Experimental in GeNie & SMILE Setup	84
7.3.3.1 Training.....	84
7.3.3.2 Testing.....	85
7.4 EXPERIMENTAL RESULTS OF AUs RECOGNITION.....	86
7.4.1 Single AUs Recognition for Image Sequences Containing Single AUs	86
7.4.2 Single AUs Recognition for Image Sequences Containing AUs combinations	87
7.4.3 AUs Combinations Recognition for Image Sequences Containing AUs Combinations	87
7.5 EXPERIMENTAL EVALUATIONS OF AUs RECOGNITION.....	89
CHAPTER 8.....	91
EMOTION RECOGNITION BY EXPERT SYSTEM	91
8.1 EXPERT SYSTEM.....	91
8.2 IMPLEMENTATION	94
8.2.1 CLIPS-Based Expert System	94
8.2.1.1 Implementation Details	94
8.2.1.2 Testing.....	96
8.3 RESULTS EVALUATION	97
8.4 SUMMARY	98
CHAPTER 9.....	101
CONCLUSIONS AND FUTURE WORK	101
9.1 CONCLUSIONS	101
9.1.1 Goals and Objectives.....	101
9.1.2 Facial Motion Measurement.....	102
9.1.3 Facial Expression Representation	103
9.2 FUTURE WORK.....	104
BIBLIOGRAPHY	107
APPENDICES A:.....	113

SIX BASIC EMOTION STATES DESCRIBED BY ACTION UNITS.....	113
APPENDICES B	119
OPTICAL FLOW-BASED FACE FEATURE REPRESENTATION FOR AUS RECOGNITION	119
APPENDICES C	121
DBN MODELING SOFTWARE & TOOLBOX	121
DBN MODELING TOOLS---GENIE.....	121
APPENDICES D	135
ANNOTATION OF COHN-KANADE DATABASE	135
APPENDICES E	139
ROI SELECTION & OPTICAL FLOWS DETECTION FROM ROI MATLAB CODE	139
APPENDICES F.....	151
AN EXAMPLE OF DEFINING AND LEARNING THE FACIAL EXPRESSION MODEL	151
APPENDICES G.....	161
A BAYESIAN APPROACH TO RECOGNISE FACIAL EXPRESSIONS USING VECTOR FLOWS	161

ACKNOWLEDGEMENT

This Master's thesis is the summary of my graduation project at the Man-Machine Interaction Group, Faculty of EEMCS, Delft University of Technology, on which I worked from June 2008 to May 2009. A major part of the work presented in this thesis was done at Philips Research High Tech Campus 36, 5656 AE Eindhoven, and the Netherlands. The work presented in this thesis is my own, but I could not have accomplished this without the help of others.

This thesis and the research that it describes could not be accomplished had it not been for the guidance and aid of multitude of individuals. Many of you have had a significant influence on me during my time at TUD, in a variety of ways, both academic and personal. To all of you (and you all know who you are) I express my sincere gratitude, and I hope that I can repay you in some small ways as I am able. I wish to acknowledge all the people who have given me encouragement and helped me in the completion of this study. I would like to single out the following people who had a major impact on me in the last few years.

I would especially like to express my appreciation and gratitude to my supervisor, Prof.dr.dr. L.J.M. Rothkrantz, for providing invaluable guidance, advice, support and criticism since my first arrival at TUD two years back. It was because of him that my master studies were so enjoyable and so intellectually rewarding. He has provided a good balance of freedom and interest, while teaching me not only how to do research, but how to write papers and give talks. I am extremely impressed by his ability to get an intuitive feel of the most difficult ideas. I hope some of his talents have rubbed off on me. His wide knowledge and great foresight have provided a good basis for the present thesis. His knowledge and instructions helped me to conquer a lot of challenges in the process of my thesis. Furthermore, I would like to thank him for giving us so many perfect and useful lectures, in his expert system and neural network courses, we learned a lot of knowledge and implementation approach which are used in the thesis.

I would also like to thank Ralph Braspenning, for taking me into his project team, introducing me to Philips Research, and for the weekly discussions on temporal reasoning that were both enjoyable and useful. He has also been a great source of information and his ideas and suggestions have really helped me in my research.

Another person who also works in Philips Research, Caifeng (Kevin) SHAN, Special thanks to him for convincing me that I should look for a research direction that I really enjoy and feel comfortable with. And thanks him for sharing the knowledge and helping me with the all kinds of technical problems and offering solutions.

My years at TUD and the MMI Lab have been priceless. I consider myself lucky to be a part of this institute and the lab. The people I have met over the two years have made the experience unique and one that I will remember forever. I am especially proud of the fact that I am a member of the group and that I got an opportunity to spend such quality time with all the people.

Special thanks go to Mirela Carmelia Popa, Dragos Datcu, and Zhenke Yang for their suggestions and ideas, not only relating to research, but in a multitude of things.

Mirela Carmelia Popa for giving many useful suggestions on my work, and for always making time for me although she is also very busy in her PHD research.

Zhenke Yang for helping me to find my way in the SMILE jungle and for implementing my graphical user interface design in GeNIe and for his everlasting patience when I pushed his SMILearn library to the limits.

Dragos Datcu for helping me find a direction to look for good parameters as the input data and giving valuable input on my search for a good application.

Thank you to my other friends in Netherlands: Haiyan, Iulia, Zhijie, Yuyin, Bart.

Thank you to Bart and Iulia, for the best days in last year to finish ES assignments together. Thank you for being such a great co-worker and a friend. Thank you for sharing so many unforgettable moments that I will never forget. I look forward to working with you again.

Very special thanks to Haiyan, my much closed friend. Thank you for always understanding me, for being so patient with my lack of time, for listening to me, for cheering me up in the hardest moments and always believing in me. Thank you for being my family in Netherlands.

Thank you to Yuyin, for sharing with me the secret of a spiritual life in the materialistic world of business. And thank you to Zhijie for being so patient, sweet, kind and supportive always.

Chapter 1

Introduction

Emotions are an important and interesting part of our daily life. We show different emotions in our daily rational communication and decision making. A lot of definitions of emotion have been given in previous years. In fact, there is little agreement about a definition of emotion though many theories of emotion have been proposed. Generally speaking, emotions are short-term, whereas moods are long-term, and temperaments or personalities are very long-term. In our literature report some historically famous concepts of emotion and common problems of judging emotion in recently daily life have been presented.

Even if some of the more complex emotions are specific to particular cultures, there will be certain less complex, more primitive emotions, such as fear and anger perhaps, which are to be found everywhere. In psychology, this has often involved the claim that there are basic emotions. The idea of basic emotions is that our concepts of emotions are organized hierarchically, with the non-basic emotions falling under one or more of the basic emotions. So, for instance, if we say anger is a basic emotion then we can get the less basic species of anger might be fury, rage, indignation, annoyance and so forth. Another example is jealousy might include fear and anger. According to the above opinion, we want to state that basic emotions are themselves common to all humans, and other sorts of emotion need not be. Now, the basic emotion theory proposed by Ekman has been criticized because of various conceptual and methodological problems. Figure 1.1 shows the six basic facial expressions which are anger, fear, disgust, surprise, happiness, and sadness.



Figure 1.1: Six basic emotion expressions.

Many historical theories about emotion have been developed. Despite the many theories, it is evident that people display these expressions to various degrees. One frequently studied task is the judgment of emotions—how well human observers can label the emotional expressions of others, in speech, on the face, nonverbal body language, etc? Related questions are: Do these represent their true emotions? Is there any ground-truth? Can they be convincingly portrayed? How well can people conceal their emotions? People from different regions may have different cultures that will influence the type of expressing emotion and they may express emotion using different languages. Different social settings also can influence people to express emotion. A lot of variations may simply reflect inconsistent procedures or interpretation of emotion categories, or differences between real and simulated data. Others, though, seem likely to reflect real differences in the vocal expression of emotion, from speaker to speaker, from culture to culture, and across genders and situations. Comparisons research between languages and cultures are limited so far, but the differences are really substantial. Let us show some examples, in Japanese society, an open display of emotion may be considered anti-social or “selfish” behavior, and it is considered normal to show a smile instead of anger or embarrassment. There is another example for happiness. In European, if people are really happy they will laugh and probably laugh loudly but in China even if we are really happy maybe we just smile and ladies always use their hands to cover their mouths when they smile. Emotion recognition has far-reaching potential in realistic world. Already, useful applications have been designed, built, and commercialized, and much research continues in hopes of extending this success.

1.1 Problem Description

The field of emotion recognition can be applied to consumer market, it is important to know that users are satisfied and to sell products it is necessary to know who has positive interest. For this research goal, Philips wants to categorize the emotion only to passive/active class and negative/positive because for a company they just need to know whether the consumer likes or dislikes a product. If at the first step the emotion can be recognized as passive then they will not continue to recognize what is the emotion exactly. If the emotion can be recognized as active, that is to say the person has feeling about this product then they will continue to recognize whether the emotion is negative or positive. For my thesis we will still focus on the basic emotions : “happiness”, “sadness”, “surprise”, “anger”, “disgust” and “fear” which are only six universal and emotions.

Emotions can be recognized from speech, facial expressions and other nonverbal behaviors. Information from different modalities can be used to reduce ambiguity, or to complete information. Moreover, studies from psychology show the need to consider the integration of different behaviors modalities in the human-human communication.

A natural ideal two-way interaction between the machine and the human contains not only one input signal. Generally speaking one of the inputs to the computer is a visual signal, from which gaze, posture, gestures, head movements and facial expressions. But in real life it is not enough to communicate only through visual signals. So recognizing speech is important for man and

machine interaction. On the output side, the computer may appear in the form of an “agent”—a computer-animated face or a personified animated character.

This “agent” can speak to the human through a synthesized voice and display corresponding facial and mouth movements on the screen. Even if they are not explicitly presented in the Figure, some other modalities such as tactile or physiological signals can also be used in conjunction with the video and audio signals [32].

The emotional state of a user can be covered by facial expression, speech, body language etc. The focus of this thesis is on automatic recognition of emotional facial expression.

1.2 Proposed Solution

In general, we have to analyze the fusion from facial expressions, head movement, gaze direction and upper-body and analysis speech signals fusion from tone, marked voice and words. In addition, the system should employ the tightly coupled audio and visual modalities; moreover, the multimodal signals should be considered mutually dependent rather than be combined only at the end as is the case in decision-level fusion.

Now neither feature-level or decision-level are an ideal approach. Model-level fusion becomes a mainly and challenging issue that brings benefits from feature-level and decision level, for this research goal there are many issues which require further investigation [33].

The goal of the thesis is to recognize emotions from facial expression that is a common approach is to localize special points on the contour of eyes, mouth and eye brows, but the localization of points are far from trivial. We will use another approach that is to localize special regions and facial expressions are generated by direction or dilatation of facial muscles. The result of muscle movement is visual as vector flow (feature).

1.3 Thesis Motivation

Note that although the focus of the current research is on emotions, this does not imply that other types of affective states are irrelevant for product experience. Obviously, our emotions are influenced by our moods. For instance, in the same way, a person’s emotional response to products may vary depending on his mood. Consumer researchers found that moods have a strong influence on consumer behavior. Somebody who is cheerful will be attracted to products more readily than someone who is in a bad mood. Conversely, emotions also influence our moods. It is not difficult to imagine that our emotional traits can also influence our emotional responses towards products. Someone who is known for his grumpy character will probably more often experience dissatisfaction with products than someone with a cheerful character. So in human-computer interaction field and for consumer market to find relationships between product appearance and emotions is the most important research aim in current research.

Facial expressions are generated by activation or relaxation of facial muscle. The visual results of muscle activity are AUs (basic movements in the face). The research presented in this thesis provides a framework on how to apply Bayesian Network for modeling the processes of recognizing some important AUs related to the six basic facial expressions separately. First the Bayesian Network structure which prescribes the exact relations between the variables was created in GeNIe which is a reasoning software solution developed at the Decision Systems Laboratory of the University of Pittsburgh. The implementation enabled us to use this software for creating a Bayesian Network which contains temporal reasoning that provides a temporal plate, and anchor, co temporal and terminal variables. Moreover, to the best of our knowledge it is the first implementation of training dataset using a temporal and inference model and algorithms.

Furthermore, the BN model was implemented and validated in GeNIe & SMILE toolbox. We present our methodology for obtaining the BN structure and learning the parameters for our extension of the BN formalism, including completing Confidential Probability Tables (CPTs). Finally, we test our BN model by the using Cohn-Kanada database of facial expression.

In this case we were able to make a good model, because we know the underlying relations between six basic emotion and related AUs. However, the problem that we are interested in is the situation where AUs were recognized in related Region of Interest (ROI) and considered the complexity of our model we should choose those ROI which contains the most complex motions for six basic emotions separately instead of using all of ROI together. The second challenge is that we cannot use the features detected from optical flow algorithm directly, that is to say the exact parameters are unknown and cannot be measured. We must find out the parameters and measure them.

1.4 Thesis Overview

The thesis is part of the requirements for Master of Science degree at the Man-Machine Interaction Group within the Faculty of EEMCS of Delft University of Technology. This thesis is a result of research done at the Man-Machine Interaction Lab in TUD in collaboration with Philips Research in Eindhoven.

In this thesis, we propose to use a BN to model and learn such relationships between motion of vector flows and AUs. A BN is a directed acyclic graph (DAG) that represents a joint probability distribution among a set of variables. In a BN, variables are represented by nodes and conditional dependencies among variables are denoted by the links among nodes. The dependency is characterized by a conditional probability table (CPT) for each node [7].

We derived an initial BN structure by analyzing the AU in FACS-coded images from the Cohn Kanade DFAT-504 database which is a facial expression database [32].

The relationships between vector flows motion and the important 18 AUs are separately learned from the human coding-optical flow tables.

The thesis is globally separated in five parts:

- Part I chapter 1 contains an introduction about multimodal emotion recognition; the related work and research are given in chapter 2; a short overview of related work in the practical field is reported in chapter 3; the main technology and current research method is presented in chapter 4, the main contributions in this thesis, the research goal and research architecture in chapter 5. In this part we describe the preliminaries or framework that this research is performed in.
- Part II chapter 6 contains our research architecture in chapter 6; our design and implementation of temporal reasoning in the GeNIe & SMILE software described in chapter 7; and our methodology of building the BN model; estimate the BN parameters and the discrimination process in chapter 8. In this part we implement the framework described in part I and created a BN model to recognize AUs. It contains general work on BN theory and application, which we will apply to specific problems in part III.
- Part III contains our experiments and experiment evaluation of recognizing AUs by the BN model. We present four experimental designs and evaluate the results for the four approaches.
- Part IV chapter 12 contains the conclusions and the possibilities for future work following from this research.
- Finally, part V contains the appendices. Appendix A contains the GeNIe tutorial that will be included in the GeNIe documentation; appendix B contains a emotion expression board and appendix C contains the parameters we used to estimate the detected features and appendix D contains all recognition results and some program code.

1.5 Thesis Contributions

The following constitutes the framework based on which our approach is improved and it offers significant advantages over the techniques previously addressed in this field.

- **Feature Detection:** Most attempts on the representation of visual information for facial expression have focused on optical flow analysis from facial action by K. Mase(1997), Y.Yacoob(1996), I.A.Essa(1997), and J.F.Cohn(1998). In their research the optical flow is used to either model muscle activities or estimate the displacements of feature points. In our thesis we defined nine Regions of Interest (ROI) which are related the AUs representation. By computing optical flow we can find which regions are changed most obviously for the six basic emotions. So we will not consider all of nine regions for every emotion because of reducing the complexity of our model.
- **Model:** In previous work, Lien [2] explored Hidden Markov Models (HMMs) for facial AU recognition and Tian [5] presented a Neural Network (NN) based approach, in which two separate NNs are constructed for the upper face AUs and the lower face AUs. But the disadvantages are obvious, the HMM can model uncertainties and time series, but it lacks the ability to represent induced and nontransitive dependencies. However, a facial

expression consists of not only its temporal information and transient cues. Other methods, e.g., Neural Networks (NNs), lack the sufficient expressive power to capture the dependencies, uncertainties, and temporal behaviors exhibited by facial expressions. Considered the limits mentioned above we decided to use BN. From psychological views facial expression representation integrates the dynamic Bayesian networks with the facial AUs. The Bayesian Networks (BNs) provide a coherent and unified hierarchical probabilistic framework to represent not only the probabilistic relations of facial expressions to the complex combination of facial AUs, but also temporal behaviors of facial expressions [28].

The main contributions of this work are as follows: first, in order to recognize AUs that are related to six facial expressions we defined nine regions of interest in which we proposed various facial feature measures to related AUs. The second property of this approach lies in the explicit modeling of the dynamic and stochastic behaviors of facial actions and the relations between AUs and related regions of interest. The main works in my thesis focus on recognizing facial AUs rather than facial expressions. Though chapter 7 we recognize emotions by an expert system.

Chapter 2

Overview of Related Work

2.1 Facial Feature Measurement

In this chapter we will give an overview of related work to our research.

In previous research the most recent facial expression classification tool is FACS (Facial Action Coding System) by Ekman, Friesen and Hager [54], which is intended for describing all visually detectable changes on the face produced by facial muscle activity. The FACS system is used by human observers, after extensive training, to recognize and classify subtle facial actions [55]. Facial actions are described with objective and emotion-independent AUs, depicted with abbreviations such as AU1, AU4 or AU1+4. An evaluation study has shown that AU coding with FACS has good to excellent inter-observer reliability [55].

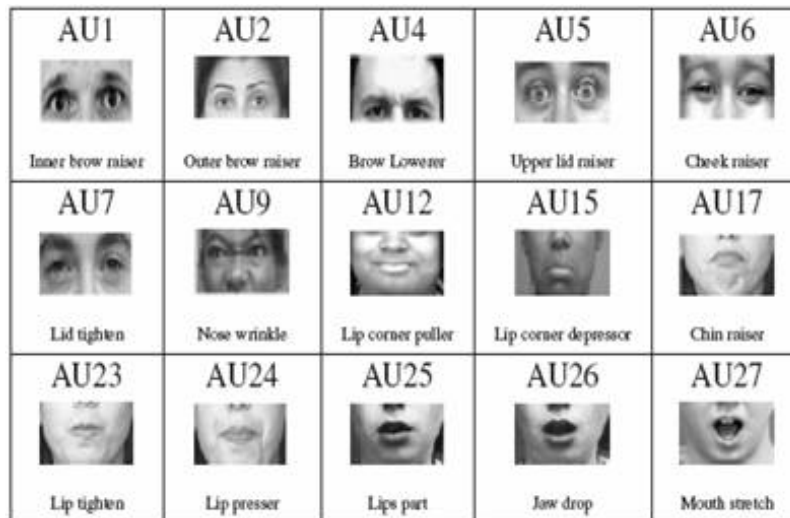


Figure 2.1: Examples of some AUs extracted from Cohn and Kanade's database.

In psychological research emotional facial expressions are coded by AUs. In order to have access to this research, it is necessary to recognize AUs in an automatic way.

Most researchers define a specific model. Then this model should be converted to AU-based model which is universal. Even in multimodal emotion recognition research field, for facial expression the AUs and AU combinations are still the most important features to recognize emotions.

2.2 Classification

A lot of facial expression classifications were studied in previous work such as the research group of Cohen et al. In addition, multimodal emotion recognition has been explored in recent years. The groups use feature-level and decision-level to combine different channels but the limitations cannot be ignored. For recognizing facial expression by AUs some groups use Gaussian Tree-Augmented Naive Bayes (TAN) to learn the dependencies among different facial motion features in order to classify facial expressions [56]. However, due to TAN's structure limitations, it is not good to handle complex relationships between facial features, as well as temporal changes. Zhang and Ji exploit a BN to classify six basic facial expressions with a dynamic fusion strategy [56]. Lien et al. [57] explored HMMs for facial AU recognition. Tian et al. [58] presented a NN-based approach, in which two separate NNs are used. Two sets of AUs which are upper face AUs and lower face AUs are considered in NN construction. GU and Ji use a similar idea for facial event classification such as fatigue [56]. Cohen et al. [59] further propose a classification driven stochastic structure search (SSS) algorithm for learning a BN classifier to recognize facial expressions from both labeled and unlabeled data.

In recent works, the emotion recognition is not only based on the signal channel but also on body gesture, speech or even head motion and gaze signals. Rana El Kaliouby and Peter Robinson [60] designed a Dynamic Bayesian to model the unfolding of head and facial displays, and corresponding mental states over time. A video stream is abstracted spatially into head pitch, yaw and roll actions, and lips, mouth and eyebrow actions. The actions are in turn abstracted into displays and mental states. The displays presented in a model of a mental state are determined by a feature selection mechanism. Figure 2.2 shows the relation of different extraction of head and facial actions [60].

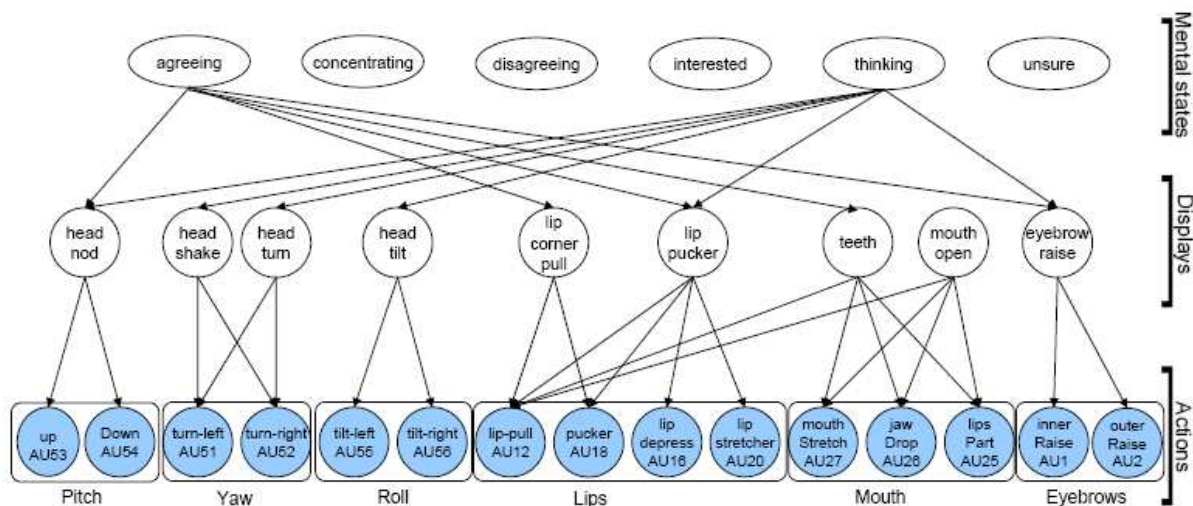


Figure 2.2: A model of a mental state determined by features extraction of Head and Facial Actions.

A video stream is abstracted spatially into head pitch, yaw and roll actions, and lips, mouth and eyebrow actions. The actions are in turn abstracted into displays and mental states. The displays

present in a model of a mental state are determined by a feature selection mechanism. For clarity, the displays for only two mental states are shown [60].

Kawato and Ohya have described a system to detect head nods and head shakes in real time by directly detecting and tracking the “between-eyes” region. The “between-eyes” region is detected and tracked using a “circle frequency filter”, which is a discrete Fourier transform of points lying on a circle, together with skin color information and templates [61]. Head nods and head shakes are detected based on pre-defined rules applied to the positions of “between-eyes” in consecutive frames. The following Figure (Figure 2.3) is the structure of the detector system.

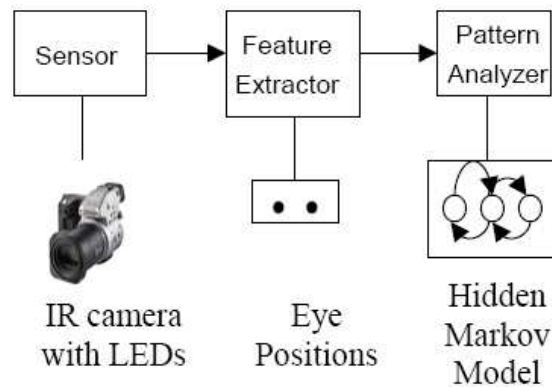


Figure 2.3: A structure to detect head nods and head shakes.

Many approaches and classifiers are used, however, in general, in the previous BN-based facial expression recognition; AUs are modeled as hidden nodes connecting facial features and facial expressions and are not recognized explicitly in their models. In addition, in fact if we just use AUs to recognize facial expression the relationships among AUs cannot be ignored. These questions and the limitations existed in previous research were considered in this thesis and also became the motivation for this thesis.

2.3 Facial Expression Recognition

Most attempts on the representation of visual information for facial expression have focused on optical flow analysis from facial action [56] [65] [58] [67], where optical flow is used to either model muscle activities or estimate the displacements of feature points. In this thesis optical flow is used to estimate the motion of features in defined ROI. Lien et al. [58] explored HMMs for facial AU recognition. In his work since each AU or AU combination associates with one HMM, the approach is infeasible for covering a great number of potential AU combinations involved in facial expressions. In this thesis we first created a BN model to represent the causal relations between the ROI and facial AUs then extend the BN to a DBN model for modelling the dynamic behaviours of facial expression in frame sequences. Cohn et al. [65] presented a Bayesian probabilistic approach to recognizing the face and facial expression but in this thesis the works focus on recognizing facial AUs instead of facial expression.

The most recent facial expression classification tool is FACS (Facial Action Coding System) by Ekman, Cohn, Friesen and Hager [62], which is intended for describing all visually detectable changes on the face produced by facial muscle activity. The FACS system is used by human observers, after extensive training, to recognize and classify subtle facial actions. Facial actions are described with objective and emotion-independent AUs [63]. All of AUs are related to different parts of the faces so in the thesis, firstly, we decided the region of interested according to some AUs that are closely related to the six basic facial expressions through implementing Matlab code. In every ROI the features were detected by optical flow algorithm, and then parameters of optical flow were computed by using of another algorithm implemented in Matlab.

2.4 Fusion in Emotion Multimodal Recognition

Several works have discussed multimodal fusion but actually most of the works just discussed to combine two models to recognize emotion, which just means bimodal recognition. In general, modality fusion is to integrate all incoming single modalities into a combined single representation [64]. Typically, fusion is either done at a feature-level or deferred to the decision-level [64]. All these researches have done is to make the fusion issue tractable the individual modalities are usually assumed independent of each other. The following gives some description about both approaches: feature-level and decision-level.

2.4.1 Feature-Level Fusion

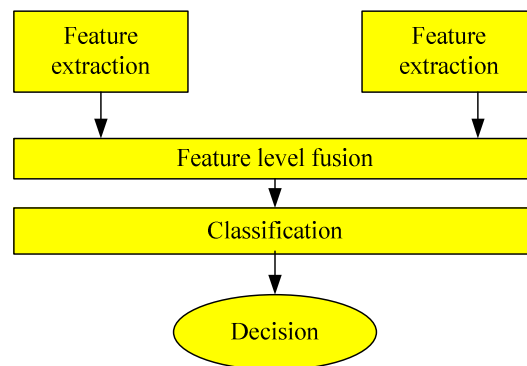


Figure 2.4: Feature level fusion structure.

Feature-level fusion (Figure 2.4) is performed by using the extracted features from each modality and concatenating these features into one large vector. The resulting vector is input to a single classifier which uses the combined information to assign the testing samples into appropriate classes. That is to say, mix together the features outputted by different signal processors. Features must be classified correctly in order to provide satisfactory results. For example features can be the position of some feature points extracted from a video processor and the prosodic features of a speech signal. This approach guarantees for multimodal fusion a good amount of exploited information but it has some drawbacks. Combining at the feature level needs

synchronization and it is more difficult and computationally intense than fusing at the decision level since the number of features is more important and features may have very different natures (e.g. distances and times). In general HMM or time biased NN are used to fuse at the feature level. In multimodal fusion, feature level fusion can almost always be applied. It can be used to mix information about voice and lip movements for speech recognition or voice and video features for emotion (expression) recognition.

2.4.2 Decision-Level Fusion

Decision-level (late) fusion is the most common way of integrating asynchronous but temporally correlated modalities [64]. Each modality is first pre-classified independently and the final classification is based on the fusion of the outputs from the different modalities. Combining information at the decision level does not mean mixing together features or signals but directly the extracted semantic information. This implies combining representations obtained from different systems that may also be correlated just at the semantic level (e.g. positions of object, with speech indicating them). Decision level fusion has the advantage to avoid synchronization issues and generally to use simple algorithms to be actually computed. But now designing optimal strategies for decision level fusion is still an open research issue. Various approaches have been proposed: sum rule, product rule, using weights, max/min/median rule, majority vote etc. [58].

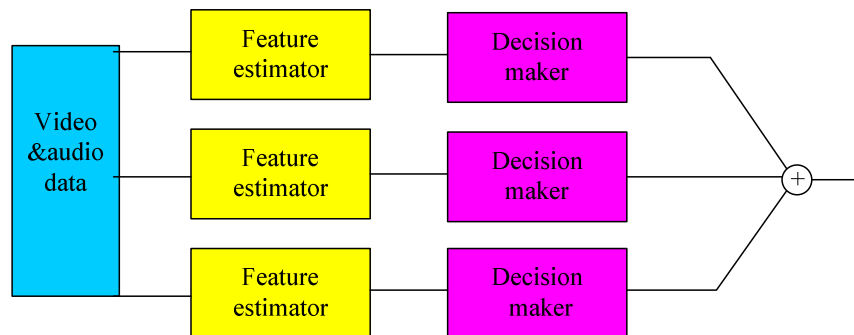


Figure 2.5: Decision level fusion structure.

Although the benefit of audio–visual bimodal fusion for affect recognition is expected from engineering and psychological perspectives, our knowledge of how humans achieve this fusion is extremely limited. It is still an unexplored issue how to construct suitable joint feature vectors composed of features from different modalities with different time scales, different metric levels and different dynamic structures. For emotion multimodal recognition research goal we think a really model-level fusion or hybrid fusion that can solve fusion problem should be paid more attention, in addition the model-level fusion should combine the benefits of both feature-level and decision-level fusion methods.

2.4.3 Model-Level Fusion

The past several decades have included a lot of advances in the field of facial expression, and then the past few years have seen growing interest in what is generally described as “speech and emotion”. People realized emotion can be communicated over various modalities including speech and language, gestures and head movements, body movement and posture. So, recent works on emotion recognition are mostly about multimodal recognition, combining all modalities to recognize emotions.

FEELTRACE is an instrument that is based on activation-evaluation space. The activation dimension measures how dynamic the emotional state is; the evaluation dimension is a global measure of the positive or negative feeling associated with the state. The FABO system includes a bimodal face and body gesture database. The goal of FABO is to add speech signals to process bimodal data and their temporal segments and fuse them at either the feature level or the decision level. However, the database from FABO is mainly about speech signals - body gesture information is rare.

A less common system is Multi-stream Fused Hidden Markov Model (MFHMM) which can be used to analyze coupled audio and visual streams to detect four cognitive states (interest, boredom, frustration and puzzlement) and the six “basic emotions”. MFHMM is a generalization of two-stream fused HMM. In this system they present a model-level fusion method named the "multi-stream fused hidden Markov model for audio-visual affect recognition". It seems that model-level fusion which combines the benefits of both feature-level and decision-level fusion methods may be the best choice for this fusion problem; however, the database is “acted”, and not natural. Ten male and female actors were asked to display facial expressions and speak appropriate sentences.

Chapter 3

Theoretical Background & Method Overview

In our literature survey, we present the theory model underlying Bayesian network and Dynamic Bayesian networks. In the method chapter we describe the theory model underlying the developed Dynamic Bayesian network. In this network there are three levels features, moreover, the higher-level features are determined by the lower-level features which means when the related lower-level features are activated the higher-level features are activated automatically.

3.1 Lucas-Kanade Method

The feature detection method chosen for the analysis of facial expressions was the optical flow technique [10, 18, 26]. This method involved assessing the magnitude and direction of facial motion. As we have mentioned before, in recent years, optical flow has emerged as a useful tool in the analysis and tracking of motion features in video sequences. Generally speaking, the traditional approach assumes the motion between two image frames at the pixel level. The motion of an object in three-dimensions (3D) is projected into a two-dimensional (2D) plane. It is assumed the type of motion sought is distributed over a sequence of several neighboring frames. In practice this assumption is reasonable as the motion is gradually spread out across several frames. [20]

The goal of our research is to analyze the activities of AUs which consist of activities of facial muscle. But sometimes the movement is really too small to be shown by vector flows. So in our case a threshold is decided which means the minimum motion mean. We only consider the motion of vectors which contain the motion mean is larger than the minimum motion mean. For this work, it was assumed there is motion of pixels across neighboring image pixels does not change over time. Therefore for two image frames the following applies [37]:

$$I(x', y', t') = I(x + \Delta x, y + \Delta y, t + \Delta t).$$

By using Taylor series expansion, small motion of pixels in two neighboring images is given by the expression:

$$I(x', y', t') = I(x, y, t) + \Delta x \frac{\partial I}{\partial x} + \Delta y \frac{\partial I}{\partial y} + \Delta t \frac{\partial I}{\partial t}.$$

The motion difference assumption results to:

$$0 = \Delta x \frac{\partial I}{\partial x} + \Delta y \frac{\partial I}{\partial y} + \Delta t \frac{\partial I}{\partial t} = \Delta x I_x + \Delta y I_y + \Delta t I_t,$$

$$u I_x + v I_y + I_t = 0; \quad u = \frac{\Delta x}{\Delta t}; \quad v = \frac{\Delta y}{\Delta t}.$$

Hence we can safely say that the optical flow expression in the u and v space is an equation of a line:

$$v = u \frac{I_x}{I_y} + \frac{I_t}{I_y}.$$

The explanation above has been reported clearly in Lucas and Kanade papers and their formulation has been used which defines motion energy functions. A Matlab program was written to implement the method.

3.2 Bayesian Network

One of the simple ways to represent a *full joint probability distribution* over all of the random variables is to design a Bayesian Network to train parameters to get the conclusion. Maybe there are a lot of parameters but Bayesian Network can use independence to decompose this distribution, reducing some parameters which have very low probabilities to present parent nodes. These properties make Bayesian networks attractive as an integrative framework for several reasons. Unlike other methods such as neural networks, Bayesian networks are not black boxes. [41] The causal relationship between variables in the network and the manner, in which a given decision was arrived at, is evident from the structure of the network [41]. It is intuitive to encode prior knowledge or parameters into the qualitative (structure), and even the quantitative (structure) parts of the network, making Bayes' nets convenient to work with which means we can use sub-models in Bayesian Network and each part will not influence other parts [42]. Finally, the probability of uncertainty with which a decision is made is encoded in the probability distribution over all possible decisions that results from Bayesian inference. In the Inference process the evidence should be taken into account, and the probability distribution over the hidden nodes should be updated, that is to say, according to the evidence and the probability with the hidden nodes to get the conclusion [13]. Essentially, sometimes not all hidden nodes have been given available evidence; a probability distribution over the possible values of these nodes can be "inferred" using the current evidence and the prior probability distribution. The result of inference is called the posterior probability distribution [42].

Bayesian network applications emphasize areas where they appear to offer a more appropriate approach than traditional computing has. Bayesian Networks offer possibilities for solving problems that require pattern recognition, pattern mapping, dealing with noisy data, pattern completion, associative look-ups, and systems that learn or adapt during use. Examples of specific areas where these types of problems appear include speech synthesis and recognition,

emotion analysis and seismic signal classification. The range of potential application is impressive.

3.3 Dynamic Bayesian Network

A Dynamic Bayesian network (DBN) is a BN extended with a temporal dimension to enable us to model dynamic systems. The tutorial in GeNIe presents an example as an introduction to basic knowledge about the DBN formalism and GeNIe interface. In this case we can learn about the difference between Bayesian Network and Dynamic Bayesian Network and how to create Dynamic Bayesian Network based on Bayesian Network. In fact, we cannot say the network structure or parameters changes dynamically though we give the temporal extension of BN, but that a dynamic system is modeled. A DBN is a directed, a-cyclic graphical model of a stochastic process. It consists of time-slices (or time-steps), with each time-slice containing its own variables [SMILE & GeNIe].

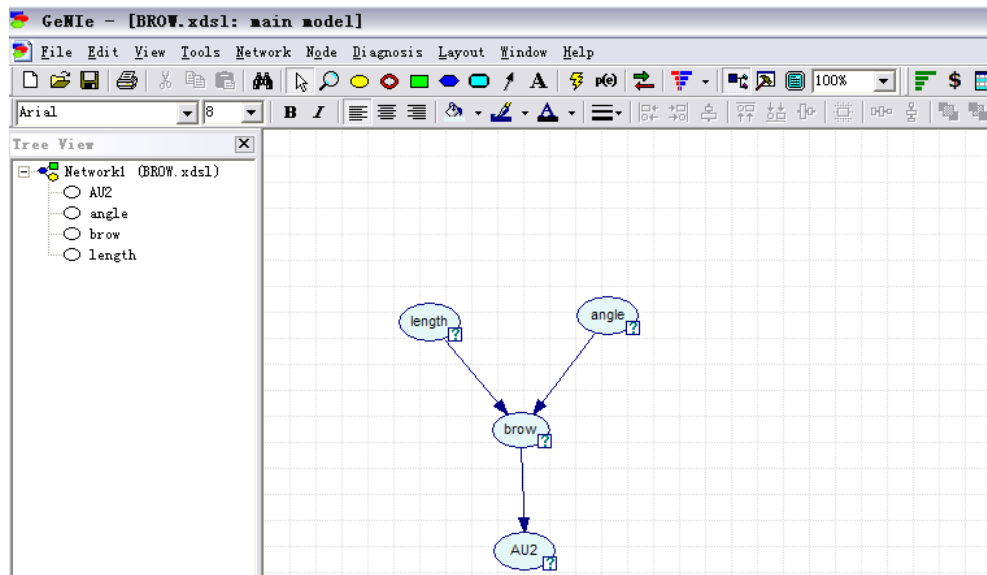


Figure 3.1: A simple example from GeNIe of a BN.

A brief background on BNs has been described in the theoretical part, the knowledge of which is helpful to understand our structure and the research goal for the project. A BN or DBN is a probabilistic belief network that is composed of variable nodes (chance, decision, deterministic) and directed arcs (essentially a directed acyclic graph) or temporal arcs (essentially a dynamic acyclic graph). Each node stands for a random variable with discrete values, and each arc was used to present a dependence (causal) relationship between two nodes. Figure 3.1 shows a simple example from GeNIe of a Bayesian network with three nodes length/angle, brow, and AU2 and two arcs going from length/angle to brow and from brow to AU2. This example will be discussed in depth later in this section.

Each node in a BN is not completed without an associated Conditional Probability Table (CPT) that presents a probability distribution over the values of the node which means its parents are applicable in this probability.

	brow	State0	State1
State0		0	1
State1		1	3.31712...

Figure 3.2: The CPT for the length/angle node.

The CPT for the length/angle node is shown in the Figure 3.2. If there are more than one child nodes for one parent node every node should be shown in the CPT and each node in the CPT represents a unique permutation of the value of the node taking the values of its parent(s). Thus, Bayesian networks consist of two distinct parts: the *structure* (nodes, arcs) and the *parameters* (prior probabilities, stored in CPTs). The structure is the qualitative part, and is relatively easy for humans to determine given its relative simplicity and the causal nature of the relationships it encodes. The parameters (prior probability distribution) are the quantitative part and are more difficult for humans to determine, although for small networks it is entirely feasible. So there are some algorithms to train input parameters to decide the CPT that is called training data in Bayesian network. When finished training data the next step is called Bayesian inference, whereby probability distributions over the hidden nodes are determined in the previous example the “length/angle” is the current evidence and the “brow” is the hidden node after training the current evidence, and both the structure and parameters of the Bayesian network to get the CPTs of the hidden nodes to get the conclusion “AU2”. Above is a simple exam to explain the Bayesian Network in the next paragraph according to our project we will present the details of modeling a DBN structure, training parameters to get CPTS and DBN inference.

In the next section, more details about modeling network, learning parameters and inference will be presented.

3.4 Probabilistic Reasoning

3.4.1 Constructing a Bayesian Network

First, consider the Bayesian theorem [13]:

- Given training data X , posteriori probability of a hypothesis H , $P(H | X)$, follows the Bayes theorem;

- Informally, this can be written as : posteriori = likelihood x prior/evidence;
- Predicts X belongs to C 1 if the probability $P(C_i | X)$ is the highest among all the $P(C_k | X)$ for all the k classes;

The Chain Rule:

$$P(x_1, x_2, \dots, x_n) = P(x_n | x_{n-1}, \dots, x_1)P(x_{n-1} | x_{n-2}, \dots, x_1) \dots P(x_2 | x_1)P(x_1).$$

So the specification of the full joint probability distribution is equivalent to the assertion that for every X_i in the network $P(X_i | X_{i-1}, \dots, X_1) = P(X_i | Parents(X_i))$ Intuitively, this says that a Bayesian network is correct only if each node is conditionally independent of its predecessors in the node ordering, given its parents. I.e., the parents of node X_i should contain all those nodes in X_1, \dots, X_{i-1} that directly influence X_i .

So the general procedure for designing a Bayesian network is as follows:

1. Add “root causes” first; then add the variables they influence, and so on.
2. Add “leaves” that are nodes that have no direct causal influence on anything else.

Note that there is an important difference between diagnostic and causal models. Causal models tend to have fewer links and the numbers are easier to come up with.

3.4.2 Steps to Follow in Constructing a Bayesian Network

1. Choose a set of relevant random variables that describe the domain.
2. Choose an ordering for the variables. Order the variables so that “root causes” are first, then the variables they influence, and so on.
3. For $i = 1$ to n , where n is the number of random variables:
 - a) Add a node to the network for variable X_i .
 - b) Set Parents (X_i) to a minimal set of nodes already in the network such that the conditional independence property is satisfied.
 - c) Define the conditional probability table (CPT) for X_i .

3.4.3 Filling in Conditional Probability Tables

It can be very difficult to fill in CPTs, but there are approaches that can help to make the process easier. The idea here is to recognize in our Bayesian network some standard pattern [19].

Deterministic nodes: A deterministic node has its value specified exactly by the values of its parents, that is to say, for deterministic nodes without uncertainty.

There are three assumptions about the model:

- Parents' nodes and child nodes are Boolean variables.
- Inhibition of one parent is independent of the inhibitions of any other parents.

According to the model we can define:

- A child node is false only if its true parents are inhibited.
- The probability of such inhibition is the product of the inhibition probabilities for each parent.
- So the probability that the child node is true is 1 minus the product of the inhibition probabilities for the true parents.

3.4.4 Exact Inference in Bayesian Networks

We can represent influence as a flow chart.

- Given: an observed event (i.e., an assignment of values to evidence variables).
- Compute: posterior probability distribution for a set of query variables.

Let X be a query variable:

$E = E_1, \dots, E_m$ is a set of evidence variables e is a particular observed event.

$Y = Y_1, \dots, Y_n$ is non-evidence variables (hidden variables).

The complete set of variables is: $X = \{X\} \cup E \cup Y$.

A typical query asks for $P(X|e)$.

- So for each value of X we compute $P(x|e) = P(x,e)/P(e)$.

We do this by computing $P(x,e) = \sum \dots \sum P(x, e, Y)$, where each summation is done over all the values of $Y_i \in Y$.

3.4.5 Bayesian Network Learning

There are four various learning options according to the structure and variables. The following presents some variations:

- Known structure, fully observable: learn CPT.
- Unknown structure, fully observable: find topology. Can hill climb using MAP or ML values?
- Known structure, hidden variables: like NNs.
- Unknown structure, hidden variables: too hard.

For our structure it is a known structure and hidden variables belief network in which the angle and length of the vector flow are observables but the regions and AUs are hidden variables. In fact, it is not very difficult for humans to find the network structure but to get the right probabilities that are really not an easy job, especially in the structure like ours in which some variables are not completely observable. In the following section we give a brief discussion about the learning problem [19] [25].

3.4.6 Learning Problem

3.4.6.1 Structure Learning

- Use greedy top-down search through the space of networks, considering adding each possible contour one at a time and picking the one that maximizes a statistical evaluation metric that measures fit to the training data [43].
- Alternative is to test all pairs of nodes to find ones that are statistically correlated and adding contours accordingly.
- Bayesian network learning requires determining the direction of causal influences.
- Special algorithms for limited graph topologies.

3.4.6.2 Parameter Learning

- If values for all variables are available during training, then parameter estimates can be directly estimated using frequency counts over the training data.
- If there are hidden variables, some form of gradient descent or Expectation Maximization (EM) must be used to estimate distributions for hidden variables.

Different methods are used in logistic discrimination and in the parameter estimation of probabilistic models. Namely, in logistic discrimination the Newton- Raphson and quasi-Newton methods are typically used, while parameters of probabilistic models are usually estimated using the EM-algorithm.

The goal of learning parameters is to find probabilities to put in the CPT to maximize the probability of the data according to the given structure and a data set which is some examples of setting of the observable variables. There are a lot of algorithms for finding the highest probability of data [44]. For our structure we will use EM algorithm which is from the gradient-descent-type approach: first write probability of dataset as a function of the CPT entries then compute how data probability changes as CPTs are moved till move CPT entries in the best direction that is to say get the highest probability. Log likelihood can be nicely used to compute derivative. A disadvantage of the EM-algorithm is that it typically converges slowly near the optimum. After the present of the general idea about EM algorithm and the concept of log likelihood in the next section more details about the algorithm and how to calculate log likelihood will be given [45].

3.4.6.3 Expectation Maximization (EM algorithm) Overview

- First, random values are selected for the parameters in the CPTs for the entire network.
- Secondly, the needed weights are computed.
- Thirdly, these weights are in turn used to estimate new CPTs.
- Then, the second step and the third step are iterated until the CPTs converge.

Expectation Maximization (EM) is an iterative algorithm that passes through the data and updates parameters of the network for each round until convergence is reached:

Before discussing the details of the E and M steps, it is important to understand the concept of the *likelihood* of parameter set, given data [45].

For each EM iteration:

1. Evaluate log likelihood of current parameter set, compare to previous (or initial) log likelihood. Stop if improvement is within the threshold.
2. E-Step: For each example, calculate and sum expected counts.
3. M-Step: Normalize expected counts to obtain new parameter set.

3.4.7 Calculating Log Likelihood

The likelihood, or log likelihood as it is in practice, of a parameter set is a measure of how well a parameter set “fits” a given dataset. Otherwise stated, it measures what is the probability when the parameter set is to generate the observed data. The EM cycle terminates when the improvement in the likelihood of the current parameter set falls below a certain threshold. To determine the likelihood of one example, one can fix the values of the evidence nodes in the full joint distribution, and then sum over all possible values of the hidden nodes [44]. For each permutation of the hidden nodes (the evidence nodes are fixed), take the product of the product, over all nodes, of the appropriate CPT cell parameters given the value of the node and its parent(s) to obtain the likelihood of that example. To find the likelihood of the entire data set given the parameter set, one must perform the above calculation for every example, and then take the product of all of these results. In most circumstances, this results in an extremely small number. It is easier to take the natural log of this expression because it reduces the product to a sum over all of the examples, which is easier to maximize and is known as the *log likelihood* [45]. Thus, the log likelihood of the data is:

$$\sum \ln \left(\prod \text{appropriate CPT parameter} \right)$$

The log likelihood is guaranteed to increase with each iteration until it reaches a *stationary point* or a threshold value, at which point the log likelihood will be a close approximation of the global maximum [45]. In order to increase the chance that the stationary point will indeed be a global maximum and not a local maximum, the initial parameters can be set to pseudorandom values rather than normalized zeroes.

3.4.8 CPTs and the Joint Probability Distribution

Have been mentioned in previous section that the problem of learning a Bayesian network is the problem of finding a network that *best fits* (according to some scoring metric) a *training dataset*. But how to decide how “well” a network matches a given training dataset that will be presented clearly in this section. Generally speaking the approach is to find network which means find both the structure of the DAG and the conditional probability tables (CPTs) associated with each node

in the DAG. Both the structure of our DBN and how to create the network in GeNIe have been described in previous section. In this section we will pay more attention in computing CPTs [46].

In a BN for every possible combination of parent states, all entry nodes should be listed in the CPT. Notice that for a large number of parents the CPT will expand drastically. A simple example in our Bayesian network is shown in right part of Figure 3.3. And the conditional probability table of node AU will have eight entries because each child node has two states (true and false) and C has 3 child nodes, with eight degrees of freedom. The CPT for node AU is given in the left part of Figure 3.3. In implementation chapter we will give more details about how to choose the parameters as input in classification and the related layers to different parameters in our model.

$P(AU VxVyAng)$	AU	\overline{AU}
$VxVyAng$	$Q_{VxVyAng}$	$1 - Q_{VxVyAng}$
$VxVy\overline{Ang}$	$Q_{VxVy\overline{Ang}}$	$1 - Q_{VxVy\overline{Ang}}$
$Vx\overline{Vy}Ang$	$Q_{Vx\overline{Vy}Ang}$	$1 - Q_{Vx\overline{Vy}Ang}$
$Vx\overline{Vy}\overline{Ang}$	$Q_{Vx\overline{Vy}\overline{Ang}}$	$1 - Q_{Vx\overline{Vy}\overline{Ang}}$
$\overline{Vx}VyAng$	$Q_{\overline{Vx}VyAng}$	$1 - Q_{\overline{Vx}VyAng}$
$\overline{Vx}Vy\overline{Ang}$	$Q_{\overline{Vx}Vy\overline{Ang}}$	$1 - Q_{\overline{Vx}Vy\overline{Ang}}$
$\overline{Vx}\overline{Vy}Ang$	$Q_{\overline{Vx}\overline{Vy}Ang}$	$1 - Q_{\overline{Vx}\overline{Vy}Ang}$
$\overline{Vx}\overline{Vy}\overline{Ang}$	$Q_{\overline{Vx}\overline{Vy}\overline{Ang}}$	$1 - Q_{\overline{Vx}\overline{Vy}\overline{Ang}}$

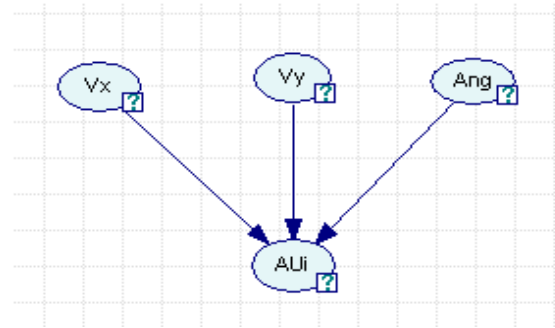


Figure 3.3: Simple example of the relation between three parameters and AUs and the CPT.

Rare event problems occur in generalized linear models, such as logistic regression, but also in models learned from data, such as neural networks and Bayesian networks.

In fact the progress of calculating of conditional probabilities is Inference in a Bayesian network according to the probabilities in a CPT. Inference is based on two rules. The first one is Bayes' rule, which is defined as:

$$P(x | y) = \frac{P(y | x)P(x)}{P(y)}$$

Informally, this can be written as: posteriori = likelihood x prior/evidence.

The second rule is the expansion rule, which is defined for binary variables ($P(x) = 1 - P(\overline{x})$) as:

$$P(x) = P(x|y)P(y) + P(x|\bar{y})P(\bar{y}),$$

$$P(x) = P(x|yz)P(yz) + P(x|\bar{y}z)P(\bar{y}z) + P(x|y\bar{z})P(y\bar{z}) + P(x|\bar{y}\bar{z})P(\bar{y}\bar{z}).$$

3.5 Expert Systems

This section presents the implementation details of an expert system. Expert systems are not a new concept but rather a specialized branch of artificial intelligence. Expert systems are essentially symbolic computer models of human expertise in a specific domain of work. An expert system is comprised of three basic components, the facts list, knowledge base and inference engine. The facts list contains the data items on which inferences are made [51].

Before getting into the intricate details of the produced solution, the utilized programming environment will be examined. Finally, the system's structure and principles are regarded and the reader is provided with the exact programming code the product is composed of.

3.5.1 Environment

This paragraph discusses the implementation environment. There are a variety of expert system shells available for the development of a system. CLIPS and Jess environment as used by the development group to produce a satisfying end result, due to our familiarity with the formatting and the availability of supporting material. Another deciding factor is stability. It is important to select a shell that has been around long enough to have been proven powerful and able to stand the test of time [52]. Furthermore, a considered alternative will be regarded.

Jess was written specifically to be compatible with CLIPS and was actually a programmer's library written in Java. Using Jess, it is possible to build or run Java applets and applications that have the capacity to reason using knowledge supplied in the form of declarative rules [54].

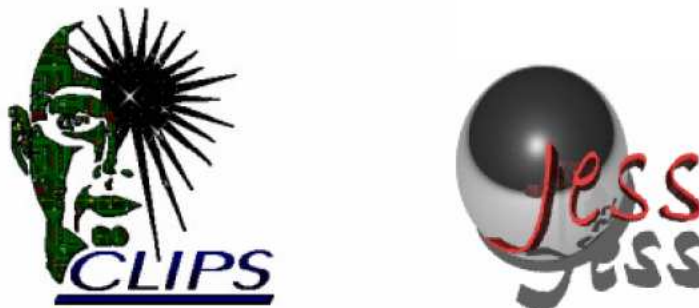


Figure 3.4: A tool for building expert systems [50] (left) and the rule engine for Java TM platform [49] (right).

Compared CLIPS to Jess, we preferred to continue working with the CLIPS programming environment. It offered sufficient technology to complete the task at hand, since the solution that

would rely more on its internal reasoning capabilities .But it was to be developed did not require an extensive user interface.

3.6 Expert System by CLIPS

The language of CLIPS is similar to LISP, as it utilizes list processing. The inference engine of CLIPS supports forward chaining [51]. It is able to represent knowledge in the form of rules, defined functions, generic functions and object-oriented programming [53].

3.6.1 Inference Engine

This section describes the general structure on which the developed expert system operates. First of all the inference engine is the driving force behind the expert system. It places rules from the knowledge base on the features for activation. Rules are activated when all patterns of a rule are matched by features. The reset command asserts the facts by bringing them into the active workspace. The run command then begins execution of the program [51] followed by a discussion of the system's algorithms, referring to the underlying programming code.

3.6.2 Knowledge Base

The knowledge base contains facts and rules. The knowledge base also contains a set of rules, that is to say each feature is internally represented using several properties: a uniquely identifying title. It is important to note that the fact list is separate from the rules. This is significant because it is possible to retain the same rules while supplying different facts. This modularity allows the expert system to be highly flexible [53].

3.6.3 Features List

The final part of the expert system is the feature list, which contains a series of features. There are two sets of feature ids positively and negatively influencing a hypothesis of the activation in these storyboards.

The system's global goal being the distinction between different activations based on a set of selected features, the fashion in which the features are interpreted has to be determined. Each feature in the list has either a neutral, positive or negative influence on the hypothesis of certain activation.

Chapter 4

Research Goal and Architecture

4.1 Research Architecture

The research goal of the thesis is to model a probabilistic and dynamic framework for facial AUs that are related to six basic emotions recognition. The classification is performed by projecting the input parameters of a test image along the optical flows. From the Figure below the two main parts of our work are parameters estimation and BN model design. In the implementation part a more detail flow will be presented which contains all steps in our research. In Figure 4.1, the main research steps in the thesis are clearly shown.

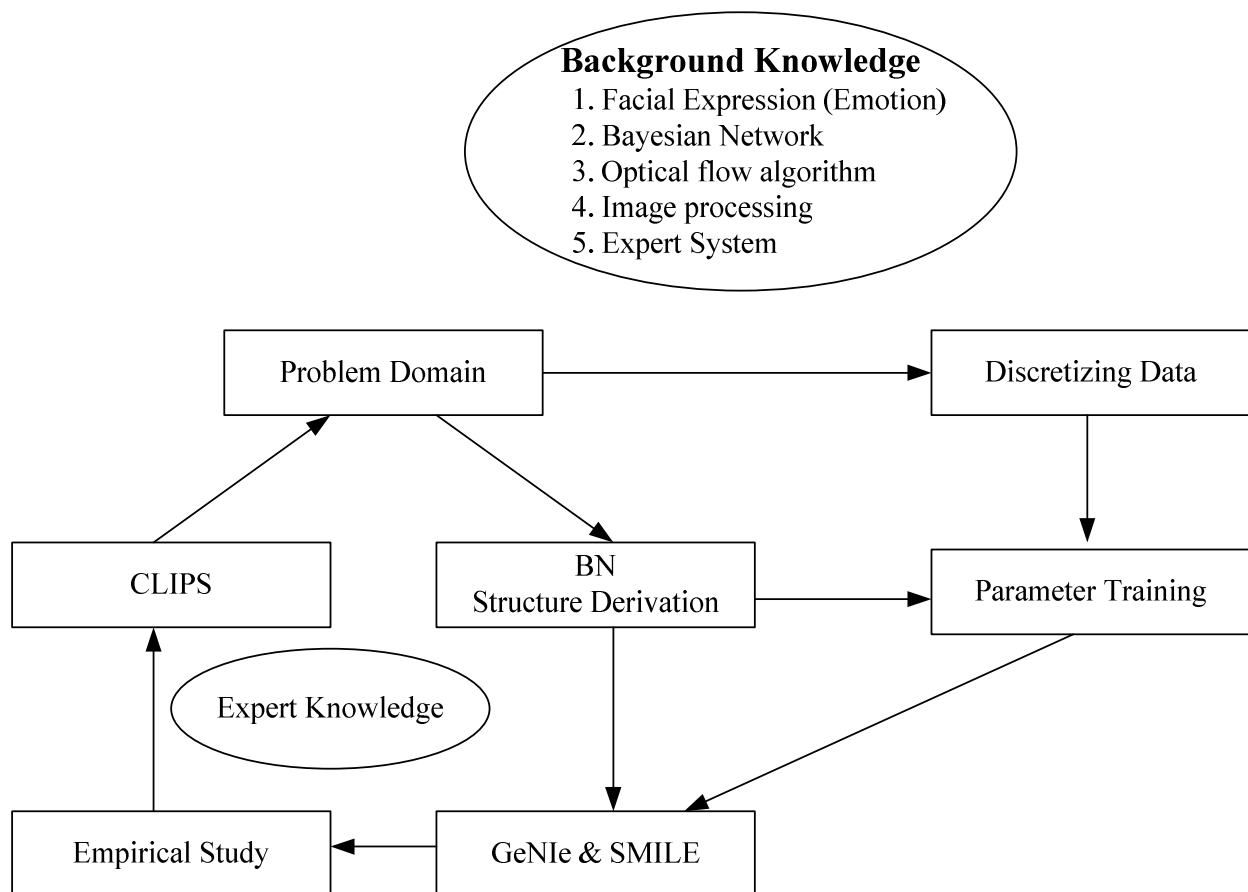


Figure 4.1: Research architecture.

Background knowledge: It is impossible to do good research without learning some knowledge of the research field. Background knowledge means the basic research station and the related subsystem or research architecture. In the thesis the background knowledge includes the facial expression domain, Bayesian network theory, Optical flow algorithm, image processing, existing software tools and related work. All these elements have been discussed in part I of the thesis which contains the related work and theoretical background.

Problem domain: The research goal of the thesis is to recognize AUs by modeling BN and recognize six basic emotions by Expert System. Two main problem domains are AUs recognition and emotion recognition. So in chapter 5 we will give the research architecture for AUs recognition domain and in chapter 7 we will use Expert System to recognize emotions by rules which are combined by AUs. An ideal situation would be gathering data from natural life but we do not have such database at our disposal so we used the data prepared from Cohn-Kanade database.

Parameter learning: It is not possible to input the feature extraction to BN structure to learning so after extracting features from database it is necessary to define the parameters as input. Some existing BN learning algorithm can be applied in GeNIe&SMILE, in our case we choose the EM algorithm. A special-purpose C++ tool was written on top of SMILE to perform parameter learning.

Discretizing data: Matlab was used to prepare data and collect dataset because it provides many mathematical and graphical representation possibilities. In our case, features should be extracted first then compute the parameters, meanwhile classify those parameters into different datasets then all datasets are labeled according to learning system.

Modeling in SMILE&GeNIe: The design of the BN causal structure should best reflect experts' understanding of the domain. According to the motion measurement by computing optical flow and the relations between AUs that are related to the six basic emotion expression categories in introduction part , we build the BN model in SMILE&GeNIe, which best represents the relationship between AUs and optical flows movements in each ROI for static face images. Our BN model of facial expression consists of three primary layers, namely, emotion expression classification layer contains 6 basic emotions, optical flow data layer contains ROI, and facial AU layer.

Empirical studies: The above subsystems are needed to perform empirical studies on modeling the problem domain with a DBN. Special-purpose tools were written in both Matlab and C++ to perform several experiments which include recognizing signal AUs and AU combinations. And all the AUs recognized will be used to combine the rules to recognize the six basic emotions in Expert System.

CLIPS: The produced expert system's internal workings are in fact quite simplistic, but to some extent it is a contribution of combining BN and ES to recognize emotions. We design the application which can output all processing steps, allowing evaluation and validation of decisions. Actually in practice, validating the application entails running the expert system, while a test user is providing the required input, and manually checking whether or not the console output matches the desired output. Because the algorithm is simple, we think a user is very well

capable of performing this final verification step. In a similar fashion, we consider any unexpected and undesired results as flaws in the underlying programming code, which are subsequently fixed.

4.2 Research Goal

The main goal of our thesis was the recognition of emotions in video streams. In this chapter we introduce the architecture of our designed system and the underlying models of the different module. According to our research goal of recognizing AUs by the extracted features which are vector flows detected by optical flow algorithm, and then all of the parameters were trained by BN. For above mentioned research goal our system is composed of several components which are displayed in Figure 4.1. The architecture figure (Figure 4.2) illustrates the main three procedures for our project and we will describe every module in more detail.

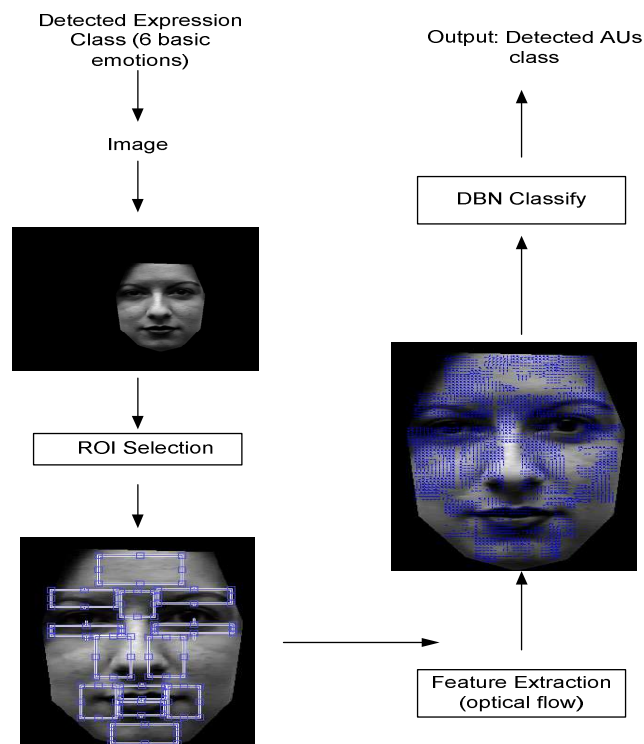


Figure 4.2: Architecture of the main three procedures.

● Region of Interest (ROI) Selection

Firstly, nine regions were defined: Brows, Lips, Lip Corners, Eyelids, Cheeks, Chin, Mouth, Nasolabial Furrow, and Wrinkles which are related some important AUs for six basic facial expressions.

Secondly, considered the complexity of our DBN model not all of those nine regions are needed for every facial expression. Entropy maxim algorithm is implemented to find out the ROI which contain most complex motion of features for six basic facial expressions separately.

- **Feature Extraction (Optical Flow)**

In this thesis we proposed a region tracking algorithm to integrate spatial and temporal information at each frame in an image sequence. The rigid and no-rigid facial motions are extracted by computing optical flow by Lucas & Kanade method. Parameters of those features were estimated and labeled for training in DBN model. After detecting vector flows for each ROI, parameters were computed as V_x, V_y and A which denote the difference between two vector position and average angle of vector flows in each ROI. Those parameters will be used as data to train in DBN model.

- **Action Units Classification**

A BN model structured was created which contains three primary layers, namely, emotion expression classification layer, optical flow data layer, and facial AU layer. And the model was implemented in BN toolbox; moreover an intuitive graphical user interface is designed to BNs in GeNIe. Implementation table for some important AUs recognition were completed for the basic facial expression separately in which the probability of each AU was recognized according to related ROI was clearly shown.

After discussing the main steps above, the main procedure of the research is clearly. The first step is to write a Matlab program according to the Cohn-Kanade data to implement the Lucas and Kanade optical flow algorithm. Before applying the algorithm to the dataset we have removed background for each frame. Because all the backgrounds in every frames are almost the same but optical flow works well when two frames have much difference (the vector flows are not requested for background).

After detecting features and computing parameters, for a classification problem, we choose BN as classifier. The mode design must be made including the topology and size of the network. The number of processing units are specified, along with the specific interconnections that the network is to have. Processing units are usually organized into distinct layers, which are either fully or partially interconnected. Next there are internal parameters that must be “turned” to optimize the model design. Finally, the selection of training data presented to network influences whether or not the network “learns” a particular task.

4.3 Research Overview

First, a Region of Interest (ROI) is established, where feature extraction will be performed. State-of-the-art techniques have used holistic methods where the ROI is the entire face, and modular or facial feature based approaches, where information is extracted from specific facial regions [26][27]. Second, dimensional reduction of the selected ROI is done by a Feature Extraction

procedure. The feature we used is optical flow which is detected by Lucas-Kanada algorithm. Finally, a trained classifier uses the extracted feature vectors from each ROI in order to assign the images to one of the training classes. Popular classifiers include Neural Networks, Hidden Markov Models, Bayesian Network and Support Vector Machines. In our project, we will focus on Dynamic Bayesian Network. This basic approach has lead researches to concentrate on fine tuning each of these steps in a decoupled manner. Here we ignore possibly exploitable dependencies between them. Basically as a human designer in this thesis we must answer each of the following questions.

1. What could be the best way in selecting ROI for a face? For example, some researchers have used holistic and facial feature regions to perform feature extraction. However, we detected some problems: not all of the vector flows are the same in different region of face so we must choose the region according to the detected vector flows.
2. How do we determine what features and what regions are relevant for AU classification? That is very import for modeling.
3. How classification is performed after extracting a set of descriptive feature vectors? For our research goal we will use DBN as the classifier in which the most important problem is the model. In the model we must try to use the limited nodes.

In order to learn the relationship between motion of vector flows and facial AUs people have used the whole range of classifiers available on their set of features such as rule-based, Bayesian networks, Neural networks, HMM, NB, k-Nearest Neighbor, etc.

In research model and technical chapter we will give more details about the three main steps considering those questions mentioned.

Chapter 5

Region of Interest and Feature Extraction

So far, we have presented the theoretical part in which a method of estimating optical flow, investigating the existing BN and DBN theory and adapting BN parameter training. From now on remembering the theoretical part and research structure and goal we will show the implementation part of the thesis. Implementation part contains (1) defining the ROI and detecting the optical flows as features, (2) finding out the ROI which contains the most complex motion for the six basic facial expressions separately, (3) estimating parameters of optical flows in each ROI containing some most complex motion for the six basic facial expression separately (4) designing and implementing BN functionality in a BN toolbox, and (5) designing an intuitive graphical user interface to BNs in GeNIe&SMILE.

5.1 Overview

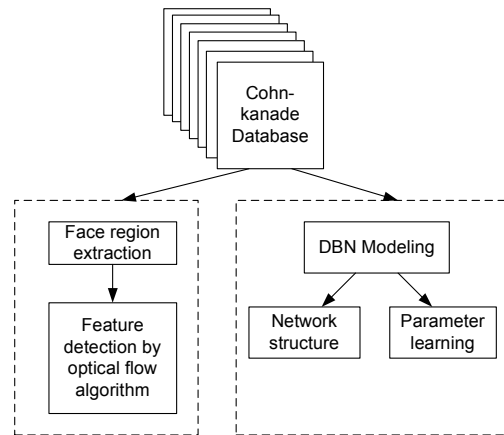


Figure 5.1: Implementation architecture.

In this section, we look more closely at some of the problems involved in taking a probabilistic approach to the management of uncertainty. First, we present the technical details of the constructed Bayesian network. In practice, this entails the presentation and discussion of the conditional probability tables and their contained values. Before providing the actual tables, we give a brief outline of the approach taken in order to determine the proper probability values. The conditional probability is defined in terms of joint features; it is the probability of the joint occurrence of one feature and another feature. Meanwhile we also give some descriptions about the main research steps in the thesis. Figure 5.1 shows the main implementation procedures in the feature extraction part and modeling part. In this chapter we mainly focus on feature extraction and modeling.

5.2 Facial Landmarks Localization

In the automatic facial expression analysis, a manual preprocessing is typically needed to select a set of characteristic points as, for example, eye centers and mouth corners, in static images or initial frame of the video sequence. These characteristic points are further used to track changes in the face or to align an input image with a face model. Currently, there is a need for a system that can automatically detect facial landmarks in the image prior to the following steps of the automatic facial expression analysis. The problem of automatic facial landmark detection has been generally addressed by modeling local texture information around landmarks and modeling shape information on spatial arrangement of the detected landmark candidates. In practice, this process consists of selecting a feature representation of facial landmarks and designing a feature detector. Different features can be detected from the image, for example, contours, colures, points, lines, and contours. These features provide a meaningful and measurable description of the face as they represent specific visual patterns which can be used to identify corresponding structures between images.

The details of the detection and orientation matching algorithm of contours of mouth, eyes, eyebrows, face, etc can be found in Yulia [40]. In Figure 5.2 are the main steps to detect all regions of interest.

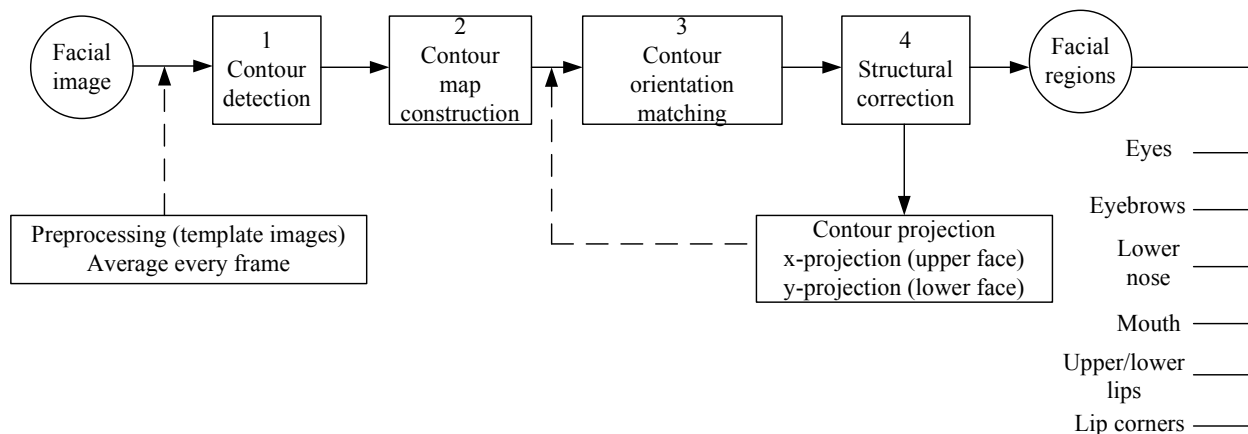


Figure 5.2: Flow chart of facial landmark localization.

5.2.1 Image Preprocessing

Cohn Kanade database consists of video recordings. Every video stream shows the recording of an emotional expression studying from neutral up to the apex target displaying with universal emotional expression.

Image sequences from neutral to target display mean duration every frames were digitized automatically into 480×3240 pixel arrays with 8-bit precision for gray scale values. The apex of every video stream was an related measure of AUs intensities, including low, medium, and high intensity.

To remove the effects of spatial variation in face position, slight rotation, and facial proportions, images must be aligned and normalized prior to analysis. Three facial feature points were manually marked in the initial image: the medial contains of both eyes and the uppermost point of the philtrum. Using an affine transformation, the images were then automatically mapped to a standard face model based on these feature points [42].

Next we have to analyze every frame by automatically controlling for face position, orientation, and magnification in this initial processing step, optical flows in each frame had exact geometric correspondence.

5.2.1.1 Image Alignment

In Cohn-Kanade database, we have image sequences from neutral to target emotions. Target emotion displays a range of AU intensities which includes low, medium, and high intensity. We mainly focus on the starting and the ending frame because both contain the most complex vector flow motion.

In the case, image alignment means making the same size of each frame. Before extracting features images must be aligned and normalized. Firstly, three facial feature points were marked in the initial image in manually. The three points are the medial canthus of both eyes and the uppermost point of the philtrum. Using an affine transformation, the images were then automatically mapped to a standard face model based on these feature points. An example of image alignment is given in Figure 5.3.

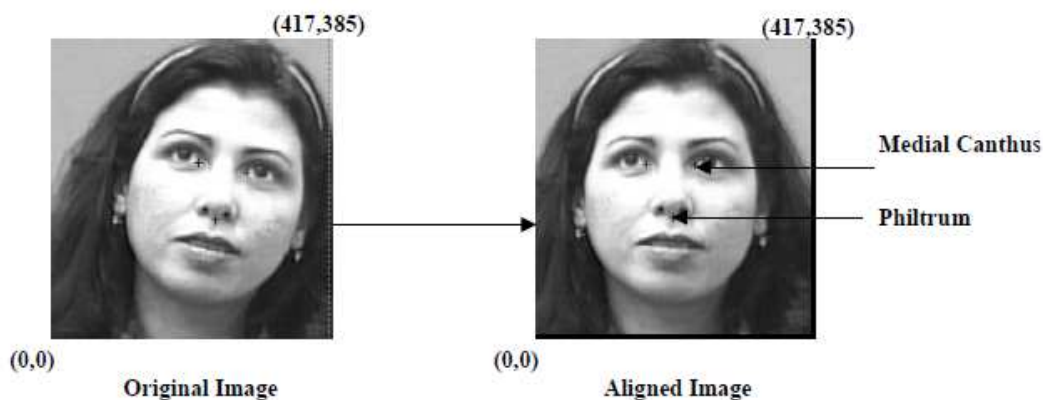


Figure 5.3: An example of image alignment.

5.2.1.2 Computing an Average Image

Generally speaking, aligned face means all images have the same size, moreover, for each frame the significant features (eyes, nose, mouth), to the degree possible, are in similar pixel locations. An example of aligned face images is given in Figure 5.4.



Figure 5.4: An example set of aligned face images.

The result of computing an average image is given in the following (Figure 5.5):

```
% Note that filenames is a cell variable (look it up!).
% Using cells is a good way to define a set of strings.
filenames = {
'4846d101.bmp'
'4848d101.bmp'
'4851d101.bmp'
'4853d101.bmp'
'4854d101.bmp'
};

number = prod(size(filenames));
image_vertical = 100;
image_horizontal = 100;
total = zeros(image_vertical, image_horizontal);

for index = 1: number
    image = read_gray(filenames{index});
    total = total + image;
    disp(index);
end

average_face = total / number;
```



Figure 5.5: The result of computing an average image.

Keep the parts of the average face that are most likely to be present in all faces exclude background and hair.

5.2.2 Algorithm of Facial Landmarks Localization

The feature based method of facial landmark localization consisted of several stages. The image was considered as a two dimensional array $I = \{b_{ij}\}$ of the size of $X \times Y$. Each b_{ij} element of the array represented b brightness of the $\{i, j\}$ image pixel. On the preprocessing stage, the image

was smoothed by the recursive Gaussian transformation (Equation 1) to remove noise and small details from the image. On the following stages of the method, the smoothed low resolution image was used to find all possible landmark candidates, and the original high resolution image was used to analyze the landmark candidates in detail.

$$b_{ij}^{(l)} = \sum_{p,q} a_{pq} b_{ij}^{l-1}, \quad (1)$$

where a_{pq} is a coefficient of the Gaussian convolution; p and q define a size of the smoothing filter, $p, q = -2 \div 2$; $i = 0 \div X - 1$; $j = 0 \div Y - 1$; l define a level of resolution ($l = 2$).

$$b_{ij} = 0.229R_{ij} + 0.587G_{ij} + 0.114B_{ij}. \quad (2)$$

On the stage of contour detection, the smoothed low resolution image was filtered with a set of ten orientation Gaussian filters (Equations 3-6) to extract local oriented contours.

$$G_{\varphi_k} = \frac{1}{Z} (G_{\varphi_k}^- - G_{\varphi_k}^+), \quad (3)$$

$$G_{\varphi_k}^- = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(p - \sigma \cos \varphi_k)^2 + (q - \sigma \sin \varphi_k)^2}{2\sigma^2}\right), \quad (4)$$

$$G_{\varphi_k}^+ = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(p + \sigma \cos \varphi_k)^2 + (q + \sigma \sin \varphi_k)^2}{2\sigma^2}\right), \quad (5)$$

$$Z = \sum (G_{\varphi_k}^- - G_{\varphi_k}^+), \quad G_{\varphi_k}^- - G_{\varphi_k}^+ > 0, \quad (6)$$

where σ is a root mean square deviation of the Gaussian distribution; φ_k is an angle of the Gaussian rotation, $\varphi_k = k \times 22.5^\circ$; $k = 2 \div 6, 10 \div 14$; $p, q = -3 \div 3$; $i = 0 \div X - 1$; $j = 0 \div Y - 1$.

The maximum response of all ten kernels (Equation 7) defined a contrast magnitude of the local contour at its pixel location. The orientation of the local contour was estimated by the orientation of the kernel that gave the maximum response.

$$g_{ij\varphi_k} = \sum_{p,q} b_{i-p, j-q}^{(l)} G_{\varphi_k}. \quad (7)$$

On the stage of contour map construction, the extracted contour points were threshold according to their contrast. The average contrast of the whole smoothed low resolution image was used to define a threshold for contrast filtering. Contour grouping was based on the neighborhood distance (D_n) between contour points and limited by a minimum number of contour points in the region (N_{\min}). Thus, contour points were grouped into one region if the distance between them was less than D_n pixels and number of contour points inside the region was bigger than N_{\min} . Regions with small number of contour points were removed. This way, the final contour map of the image consisted of regions of connected contour points presuming to contain facial landmarks. The optimal thresholds for contour grouping were determined as Neighborhood distance for contour grouping, $D_n = 1$ pixel and Minimum number of contour points in the

region, $N_{\min} = 100$ pixels. To get more detailed description of the extracted contour regions, contour detection and contour grouping were applied to high resolution image ($l = 1$) within the limits of the found contour regions. In this case, the threshold for contrast filtering was determined as a double average contrast of the high resolution image.

On the stage of contour orientation matching, the existence of facial landmarks in the image was verified. To do that, a distribution of the local oriented contours inside the located regions, so called orientation portraits, was matched against the orientation model. The model specified a characteristic distribution of the local oriented contours with maximums corresponded to two horizontal orientations (dark to light and light to dark horizontal contours). Unlike facial landmarks, noisy regions as, for example, elements of clothing and hair usually had an arbitrary distribution of the oriented contours and were discarded by the orientation model.

5.3 Region of Interest Selection and Localization

In this chapter we will give more details about facial landmarks localization and the feature extraction, moreover we will focus on the region of interest to analysis the optical flows detected from six basic facial expressions.

5.3.1 Optical Flow Regions in Face Mask

When we implement Lucas-Kanade method in two-different frames there are a lot of optical flows. But we have to find the important difference between successive faces related to emotions, i.e. muscle activations. We defined small regions which are related to length/angles of one or more muscle activations. In other words, if we know how those regions change, we will know how the facial expression changes.

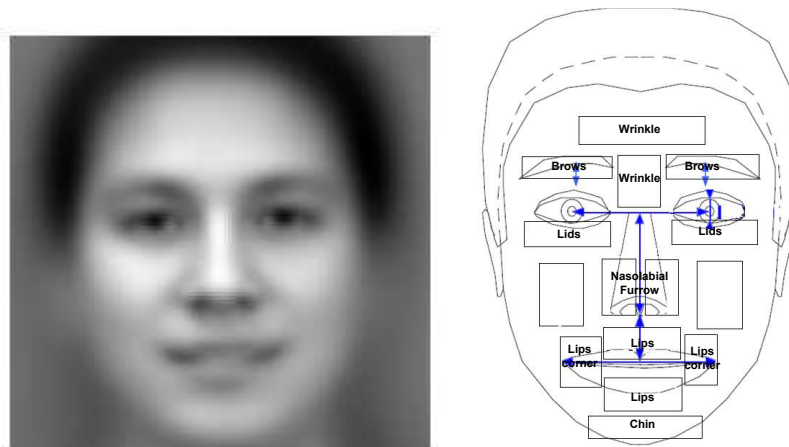


Figure 5.6: Region of interested in Mask.

As the first step in our approach we created a mask modeled a person specific surface (Figure 5.6). Feature extraction and analysis is based on the mask analysis as mentioned above. Facial motion is caused by muscle contractions. Muscle motion between successive frames is determined for each region using a version of the well-established two-frame differential method by Lukas-Kanade, which is commonly referred to as optical flow estimation [46]. In this thesis we also use the optical flow algorithm by Lukas-Kanade to detect features. After averaging all images we can evaluate the ROI in the mask.

5.3.2 Divided Regions in Face

The goal of the thesis is to recognize AUs which are related to multiple muscle structure. So it is necessary to analysis the muscle changes in the face according to each AU. All AUs are related to different parts in the face so we decide to divide regions in the face according to AUs which have to be recognized.

After implementation of the Lucas and Kanade optical flow algorithm we can get a vector space situated within the space of the image frame which represents the difference between two frames. In order to find the most obvious parts which contain the most complex motion of vector flows, we developed a system to divide the face into 20 regions and show the vectors in every region (Figure 5.7, we display 20 ROI in our system). Moreover we also developed a program to present the vector flows as a compass that depicts the vectors in a 360 degrees circular plane which will be shown later (Figure 5.14). In the histogram we can see how the vector flow changed in every region.

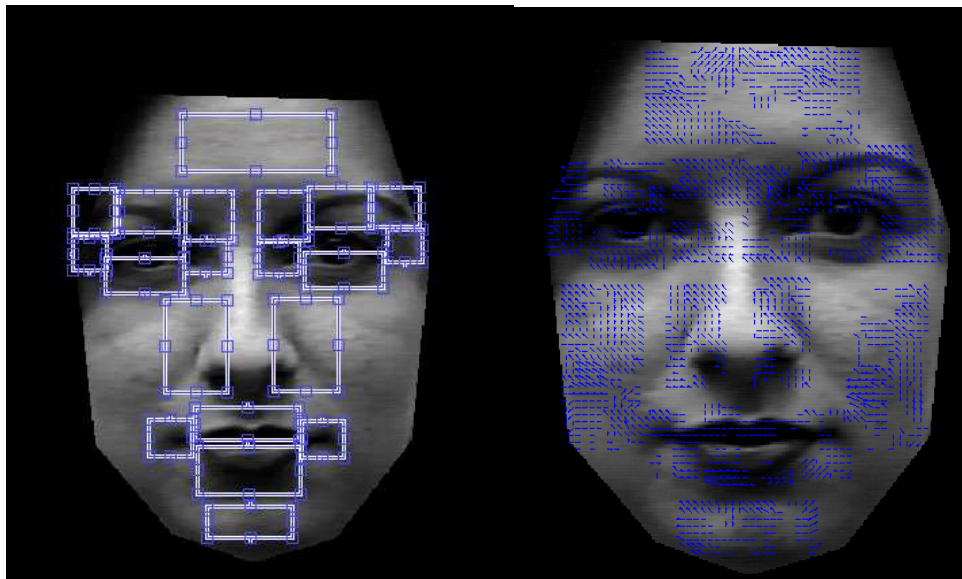


Figure 5.7: 20 interested regions related to the first step.

However, in fact not all of the 20 regions will be used in our BN model. More details and reasons will be presented in later sections.

First we validate our choice of the ROI. We focus on the 6 basic emotions, because every blended emotion is a combination of basic emotions. So if the theory holds for basic emotions we can assume that it also holds for blended emotions.

Now we can get optical flows for every region, however, a new problem raised that is whether it is necessary to use all of the 20 regions for all of six facial representations. If not, which regions should be used for each facial representation? To solve this problem we will use Entropy Maxima Algorithm to search for the regions in which the optical flows changed very obviously.

In fact, we only used a subset (nine) of ROIs instead of all of the 20 ROIs. Fortunately, the ROIs we got are almost the same as the related regions with AUs which are labeled in Cohn-Kanade database. So our computational approach corresponding with the face-evidence method of Ekman, based on human orientation. We will present the nine ROI in more detail in following sections.

5.3.3 Most Complex Motion Region (ROI) Selection

In this section our objective is to design the region of interest containing complex and fast-changing motions. These extracted regions can be used for AUs recognition, because they provide more informative motion data, which are critical for further emotion recognition. At the beginning we use optical flow analysis to measure the motion in 20 regions in a face then we only select the key frames which are key regions at the local maxima of the entropy scores, which are calculated on the histograms of motion vectors in each frame within a shot. After that Histogram Intersection is applied to measure the niter-frame difference. We want to combine the two methods to detect the region of interest.

5.3.3.1 Entropy Maximum Algorithm

The Compass Function of Vector Diagram is one of the approaches we got to analyze the motion of vector flows and in this section we will use another popular algorithm to analysis the motion of vector flows which is Entropy Maximum algorithm. Entropy is a good way of representing the impurity or unpredictability of a set of data since it is dependent on the context in which the measurement is taken.

In the proposed entropy maxim algorithm we consider to split up each facial expression as a frame sequence of 9 frames according to the 9 ROI. For example we divide the “happy” facial expression image to nine frames of Lips, Lip Corners, Eyelids, Cheeks, Chin, Mouth, Nasolabial Furrow, and Wrinkles expression. Next we distribute the entropy with respect to the frame number; we can say the frame with the highest entropy value contains the most complex motions in a frame sequence.

Our algorithm to compute the entropy is based on the optical flow. Each motion vector based on the optical flow output is quantized by its magnitude (length) and orientation (angle). Each

combination of magnitude and orientation corresponds to a bin in the vector flow compass diagram. In our implementation, 40 compass bins which represent 6 magnitude levels and 12 orientation angles are used. The probability of appearance of the K^{th} bin in a frame is given below:

$$p_f(k) = h_f(k) / M * N, \quad (1)$$

where M, N is the size of the frame and h denotes the count of optical flows in the K^{th} bin. In formula (1) the value of $p_f(k)$ is increasing as $h_f(k)$ is increasing or $M * N$ is decreasing. Fortunately, our objective is to find the region containing more optical flows.

$$E = \sum_{k=1}^{K_{\max}} e_f(k) = \sum_{k=1}^{K_{\max}} -p_f(k) * \log_2(p_f(k)), \quad (2)$$

where K_{\max} is 40 and the sum of all these entropies $e_f(k)$ is the global entropy of the motion in this frame. Algorithms introduced in this report are implemented in the Matlab workspace.

5.3.3.2 Inter-Frame Method

The entropy maximum method only provides us the information about which frames contain the most complex motions. In some situations frames in which the motion histograms change fast relatively to the surrounding frames also contain important information. Therefore, we propose another key frame extraction algorithm called the inter-frame method, which measures the differences between the motions of consecutive frames. Histogram Intersection, proposed by Swain and Ballard, is a straightforward technique to calculate the similarity between two histograms. Assume the histograms of frame and its neighborhood frame are $H_f(i)$ and $H_f(i \pm x)$ respectively, and each contains n bins. The intersection HI of two histograms as

$$HI = \frac{\sum_{i=1}^n \min(H_f(i), H_f(i \pm x))}{\sum_{i=1}^n H_f(i)}. \quad (3)$$

The denominator normalizes the histogram intersection and makes the value of the histogram intersection between 0 and 1. This value is actually proportional to the number of pixels from the current frames that have corresponding pixels of the same motion vectors in the neighborhood frame; a higher HI value indicates higher similarity between two frames. As we expected, the ideal ROI we need should contain the significantly different motion among different facial expressions.

In the following section we will show the vector flows in those regions which contain the most complex motion. In those regions the motion of vector flows are obvious and totally different from another facial expressions.

5.3.4 Automatic Localization of ROI from AU-coded Facial Images

In section 5.1 we presented the main steps and algorithm of Automatic Localization of Facial Landmarks from Expressive Images of High Complexity by Yulia Gizatdinova and Veikko Surakka [40]. In this section according to our knowledge about shape and location-ship of face parts and based on their algorithm we first implemented a feature-based method for expression-invariant localization of facial landmarks from static images. Moreover we did some improvements based on the contour map method and contour orientation matching algorithm. We applied local contrast thresholds that are calculated in every filter neighborhood instead of using an average contrast of the whole image to define thresholds for contrast filtering. Then according to the orientation of eyes, eyebrows and mouth and the knowledge of a face geometry model to compute the candidates and distance to localize the nose, lips and mouth corners. The more details of contour detection and contour orientation matching algorithm can be found in Yulia [40].

The input images are extracted facial regions using the same face localization procedure. And in this procedure we tightly couple the ROI selection step with the feature extraction process. Following we will give more details about our implementation and main steps.

Each ROI is defined with four degrees of freedom around a rectangular region: height, width, and two coordinates indicating the central pixel which are denoted as `xmin`, `ymin`, `width`, `height` separately. There are 13 structural variables $\{r_1, \dots, r_{13}\}$, represented by a single bit each. Each one controls the activation of one ROI definition block. Each ROI vector flows were loaded from related files which are computed by Lucas-Kanade optical algorithm. In each ROI block contains four parametric variables $r = [\text{xmin}, \text{ymin}, \text{width}, \text{height}]$, coded into four bit strings each. These variables define the ROI center (`xmin`, `ymin`), width and height. In essence each r establishes the position and dimension for a particular ROI.

After defining the position of each ROI all vector flows within the rectangular can be defined. Vectors which are too little to show movements a threshold should be defined (`threshold thr = 0.05`). In those ROI which contain a motion mean which is large enough, the average length and angles of each vector flows are computed.

We created an interface for this procedure. After running the program some files are created in which parametric variables computed. Those files are `\{.loc, .pctloc, .roiflowvectors, .rect\}` which stand for location of the image file, location of the pct file storing, storing vectors within the rectangles and ROI rectangles on the images. After running the procedure four structure were obtained which contain parametric variables (Δx , Δy , Len and Ang) which denote motion difference of pixels in two neighboring images and average length and angles of each vector flows of pixels .

After the implementation of the algorithm of Automatic Localization of Facial Landmarks from Expressive Images of High Complexity by Yulia Gizatdinova and Veikko Surakka [40] the

centre of eye, nose and mouth can be localized. Then there are five steps of searching ROI and the search is implemented in the upper-part of face and the lower part of face. If a landmark candidate consisted of two or more regions of contour concentration, all the contour points were departed to X-axis and Y-axis parts to landmark upper face and lower face separately from calculating the number of contour points along vertical or horizontal rows of the final contour map for the given candidate. And the local contrast thresholds are calculated in every filter neighborhood. If the number of contour points was smaller than a threshold was smaller than the threshold then the contour points were eliminated.

The search started with finding out eyes and brows using such algorithm and then according to the distance between eyes and mouth to localize nose. After finding the eyes, brows, nose and mouth the pair location is tried to find out. According to the central point and the distance between central point and contour the eye, brows, nose and mouth, the left and right parts of the face is symmetrical. In the following more details about main steps are presented.

Step 1: Finding horizontal candidate pairs with approximately equal number of contour (block) points which are four parametric variables $r = [x_{min}, y_{min}, width, height]$ and labeling them as eyes region and eyebrows region. For this step the eyes and brows are only localized as one region.

Step 2: Searching for eyebrows above and eyes below the found pair location according the central of eyeball and the dynamic parameter D which is a measurement we utilized to calculate the distance between mass central of the eye region pair. So far, the lower-lid is parted from the eye and brows region. Next step is to detected upper-lid and brows separately.

Step 3: Finding the central point that lied in the middle of central of brow and central of eyeball then use the central point and central of eyeball as parametric variables to get a block and labeling it as upper-lid region. Then the block started from the top contour of upper-lid block is labeled as brow region.

Step 4: For searching lower face region first the line of vertical symmetry was drawn through the point that lied in the middle of the line connecting right and left eye regions. Then according to the parameter D nose is localized. Nose should be located between eyes and mouth so the distance between nose and eyes was computed as $d1$ and the distance between nose and mouth was computed as $d2$. And nose is located between eyes and mouth not lower than one D from the middle point of the line connecting eye regions.

Step 5: The mouth detection also was performed from top-to-bottom along the line of vertical symmetry. And for the central point of nose and mouth the upper and lower lip can be localized separately. The mouth corners were localized by comparing the distance. The distances between central of nose and central of mouth have been computed. And when the mouth is closed the corner can be labeled far from the half of the lips. When the mouth is not only closed the distance between the corner and the nose central is not lower than the half of lip length.

Figures 5.8 (a) to (e) shows the stages of the method and were described in more detail in section 5.2. Image is courtesy of Cohn-Kanade AU-Coded Facial Expression database.

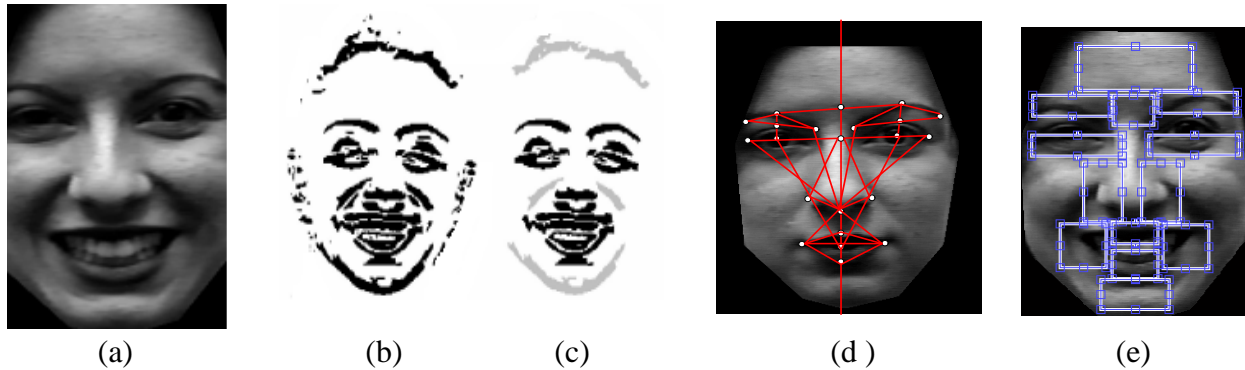


Figure 5.8: Facial landmark localization: (a) image of smoothed image; (b) extracted local oriented contours; (c) contours grouped into regions representing landmark candidates (black) and noisy regions discarded by the contour orientation model (grey); (d) face geometry model; and (e) final localization result.

5.4 Feature Extraction

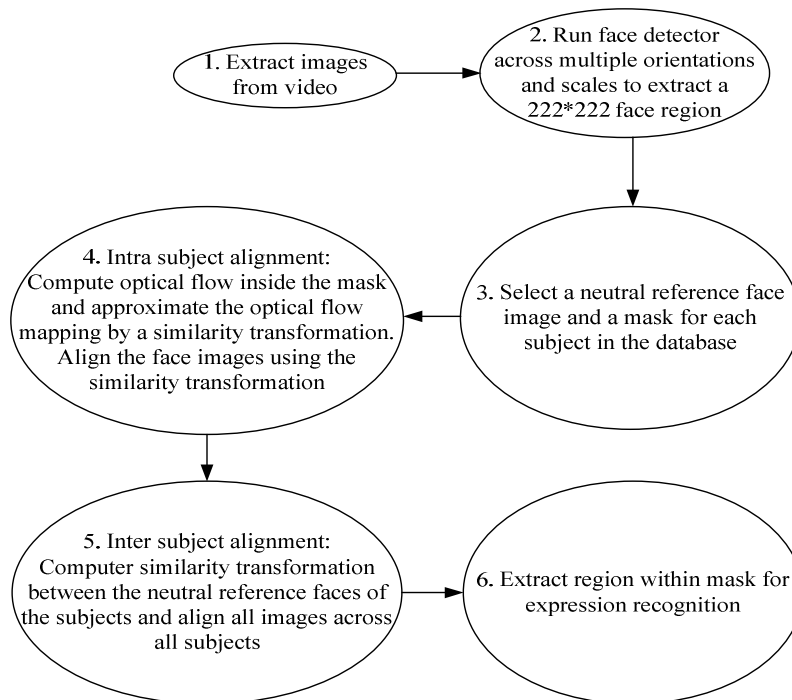


Figure 5.9: Flow chart of feature extraction.

Feature extraction involves extracting low-dimensional image features and selecting a subset of these features for classification. For this research goal we have to extract features from single frames which are the input of our BN. Such features should represent the emotional content of facial expressions. We decided to take vector flow as representatives of the emotional content of

facial expressions. From the theory of P. Ekman it is known that every facial expression can be described by AUs. But AUs are only able to describe static pictures. But the underlying idea of Ekman is that every facial expression is based on the contraction/dilatation of facial muscles. In our video frames we see the process of contracting/dilatation facial muscles [45]. We can represent a vector flow by a gradient field indicating the direction and speed of moving facial particles in the face. It is impossible to feed into DBN the whole gradient vector in sampled points in the face. We have to summarize the vector field to average values in representing parameters. We defined Regions of Interest in the length/angle of muscle movement and in every ROI we summarize the vector flow by the average direction and average speed (length of the vector). Figure 5.9 shows the main steps of extracting features.

5.5 Optical Flows Analysis

5.5.1 Optical Flow Detection and Presentation

It is not a new field in detecting face features by optical flow. Optical flow was used by robotics researchers in many Length/Angles such as: object detection and tracking, image dominant plane extraction, movement detection, robot navigation and visual odometer [26].

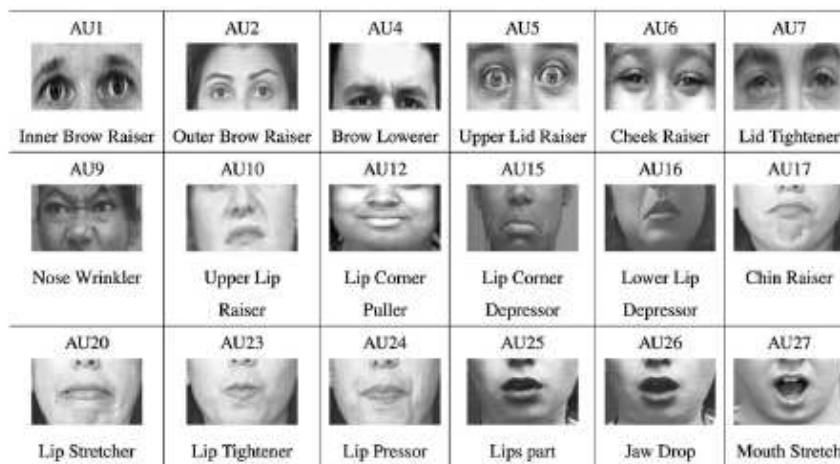


Figure 5.10: Important AUs related to the facial expressions of the six basic emotions.

There are several differential methods of estimating optical flow. For my thesis research goal, we choose Lucas–Kanade method to implement. Because his approach is based on the difference between two image frames. First the Lucas-Kanade method code was written in MATLAB and MATLAB was our implementation environment. Vector flows are encoded in compass diagrams which will be explained in following section.

Optical flow is a popular approach to define a motion descriptor to recognize motion. In this thesis we focus on recognizing the important 18 AUs which are related to six basic emotional facial expressions.



Figure 5.11: Optical flow shown in the apex of the six basic emotions.

The details are as follows. First, the sequence of optical flow fields is computed using standard approaches such as the Lucas-Kanade algorithm which has been presented above, then we compute the optical flow F of the whole sequence, being F_k the optical flow between $I_{natural}$ and I_{apex} . Each velocity vector $F_k(x, y)$ has a modulus $M_k(x, y)$ and an angle $\theta_k(x, y)$.

The optical flow corresponding to the background should be zeros since the faces are captured in Cohn-Kanade database, there are always many backgrounds. We extract the foreground movement via simple removing the highest bin in the histogram. Figure 5.11 shows the optical flow obtained between natural and apex frames.

In the following section we will give a representation of the vector flow of the six basic emotions in the face with the Region of Interest in the background. We want to research the chosen parameter average speed and direction is good representatives of the flow in the ROI.

5.5.2 Vector Flow Map and Analysis

For every vector flow in a ROI, we computed the average flow and represent this. In the compass diagram summed all the vectors from the diagram and created a determination of the total

directional vectors from the centre of the origin. It shows the major directions and velocities of facial movement.

The compass diagram (360 degrees) is sampled into 12 segments, each of 30 degrees, and represents the total directional facial movement from the neutral face image. Vectors may radiate from the centre of the compass. These data were also tabulated and the vectors summed for each 30 degree segment [47].

We computed all vectors for each 30 degree. Next we summed them. For different regions the compass vectors shown are different. In those compass we can see which degrees always frequently covered with vectors and which degrees never or rarely covered with vectors. We can conclude that for different facial expressions, the compass diagrams are different. So we may hope to recognize emotions from the vector flow in ROI.

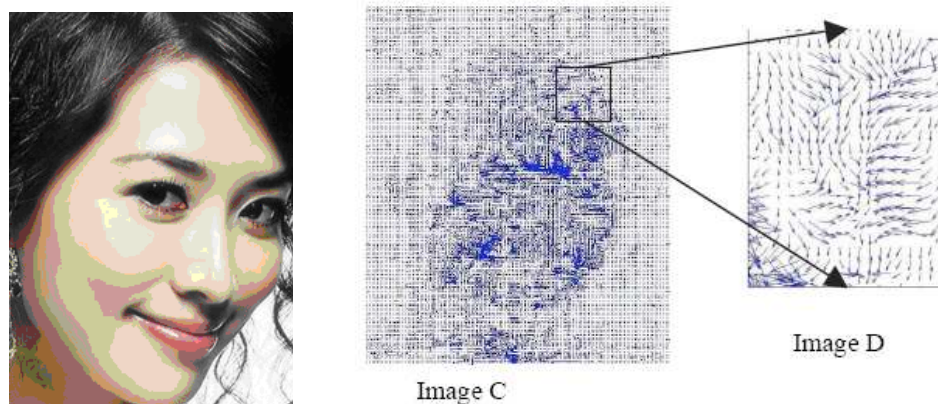


Figure 5.12: Example for Optical Flow Shown.

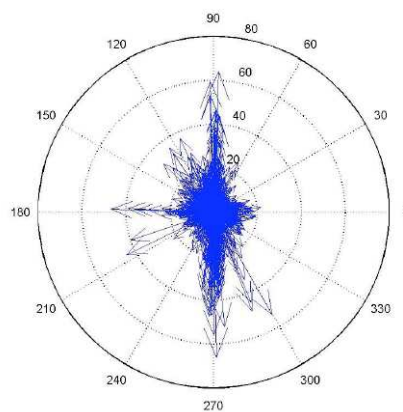


Figure 5.13: Optical Flow - Compass Function of Vector Diagram.

Optical flow analysis on each of the frames which is comprised with the natural frame in Cohn-Kanade database resulted in vector maps, an example of which is shown in Figure 5.12. The vectors (arrows) as shown in the Figure represent angles and amplitude of the summed facial movements associated with a particular emotion – “Amusement” as depicted in this map. Following we give an example of the Optical Flow Compass. For the emotion state "smile" the

motion regions are eyes and mouth corners, now we only show the compass of vector flows in the mouth corners region. Consistently produces movement in the 60-90 degrees; however, there is lack movement in the 0-90 degrees and 120-150 degrees [47].

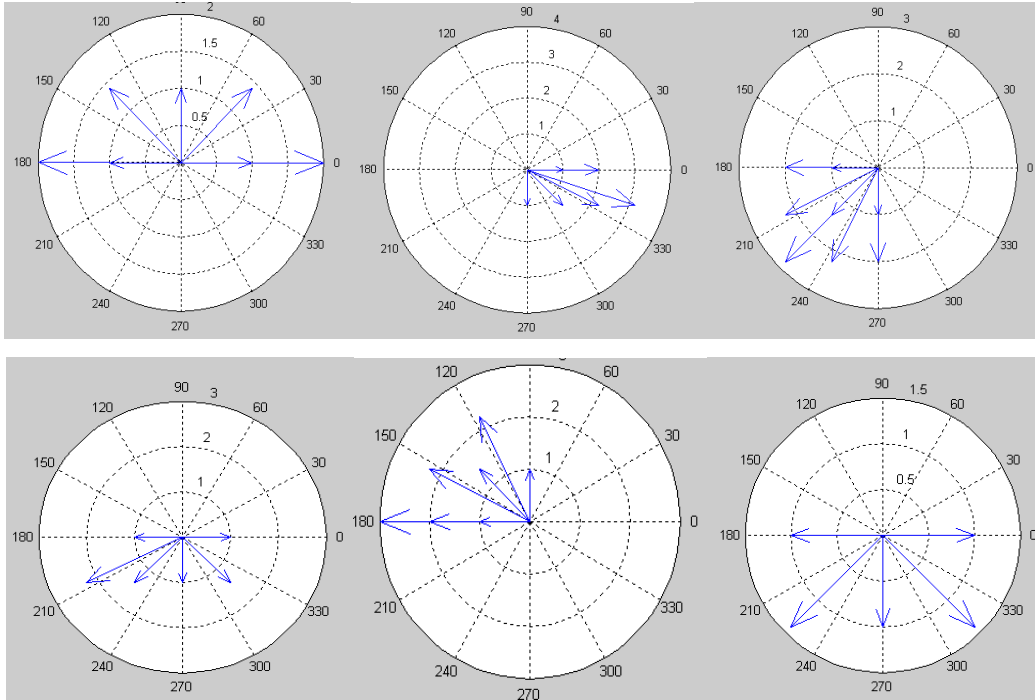


Figure 5.14: Vector flow compass in the whole for six basic emotions.

For the above example (Figure 5.14), we presented six vector compass diagrams for six basic emotions to see the difference. Because it is not easy to see the difference from the vector flows.

The example presented six vector compass diagrams for the basic six emotions are shown. From those compass diagrams we can safely say that for different emotion expressions the average of angle and length for all vectors are different. For example, comparing the compass diagrams above, it can be seen that the expression of “fear” consistently produces movement in the range (0-180 degrees), “surprise” (270-360 degrees), and “sad” (180-270degrees), “disgust” (90-180 degrees).

The vector map was converted using Matlab to a compass diagram. First step is to write a Matlab code to compile optical flow algorithm then converted those vectors to a compass diagram. The vector compass shows the major directions and velocities of facial movement. Using our system, we got optical flow compass for all regions then we can know which regions contain the most complex motion of vector flows.

For every ROI of vector flow we captured average length and average angle as parameters. Per emotion we took the average sum of the parameters in whole face. But unfortunately, the vector flow compass in whole face is not good enough to show the significant difference among different emotions. Following we present the six vector flow compass which are mapped optical

flows extracted from six basic emotions separately. So in next section we analysis the average sum of vectors of all ROI for emotion.

After comparing the compass of the vector flows in the whole face it is not difficult to find that for different facial expression the movements of optical flow are different and the average direction and intensity of main optical flows are also different. But the differences are not so significant that we have to consider whether it is good enough to recognize emotions only using three parameters. But for optical flow algorithm the parameters that we can estimate are brightness changing of neighboring image pixels (Δx , Δy), direction changing of neighboring image pixels and the length that computed from brightness changing of neighboring image pixels. Considering that each AU is related to different part in face instead of the whole face. In the next section we will analysis the vector flows in ROI for six basic emotions separately.

5.5.3 Vector Flow Analysis in Region of Interest

In the flowing Figures (Figure 5.15-5.20), we give a representation of the vector flows shown in ROI which contain the most complex motion of vector flow for the 6 basic facial expressions separately. Furthermore, for those ROI the compass diagrams also will be presented.

Anger—lower-eye lids, chin, and mouth:

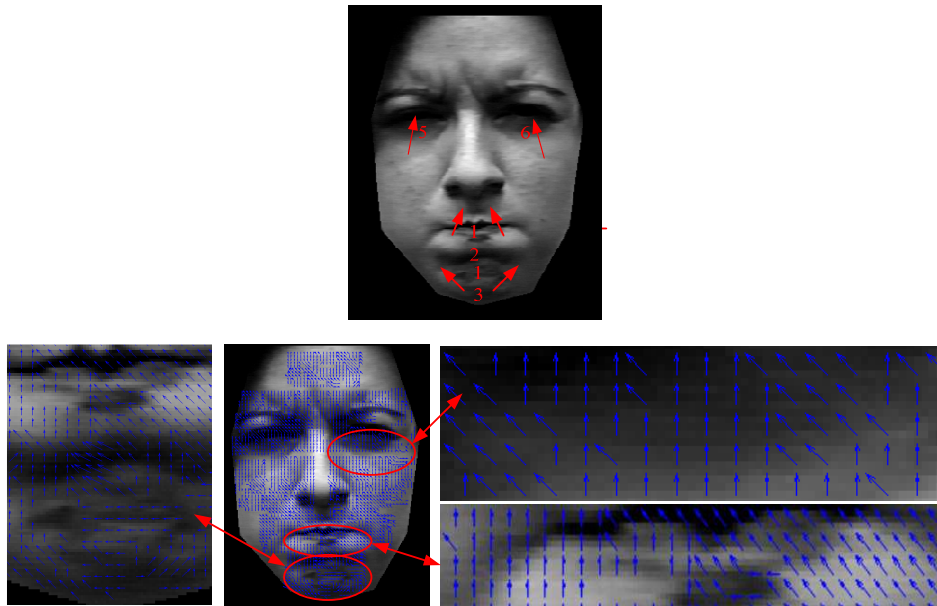


Figure 5.15: Vector Flows in ROI for facial expression of anger.

Disgust—mouth and wrinkle:

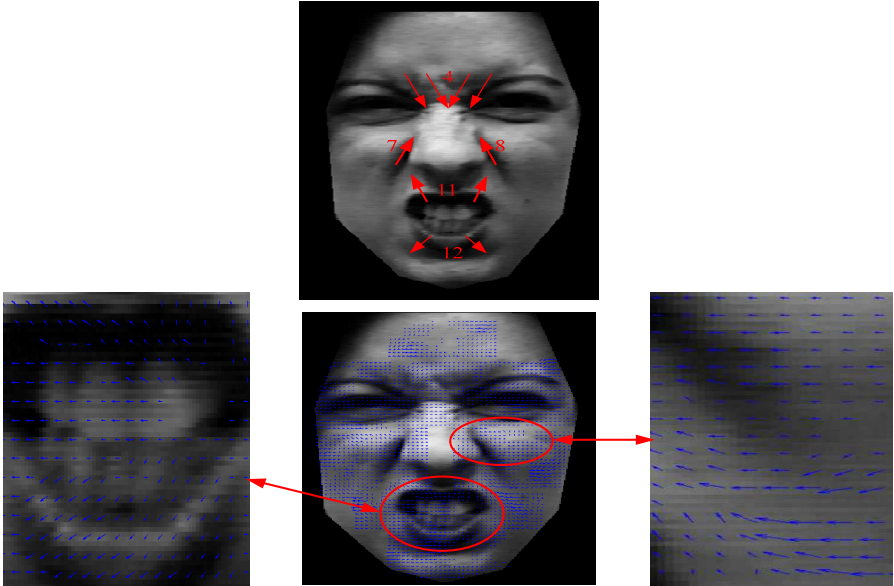


Figure 5.16: Vector Flows in ROI for facial expression of disgust.

Fear—mouth, lips corners, brows, lids:

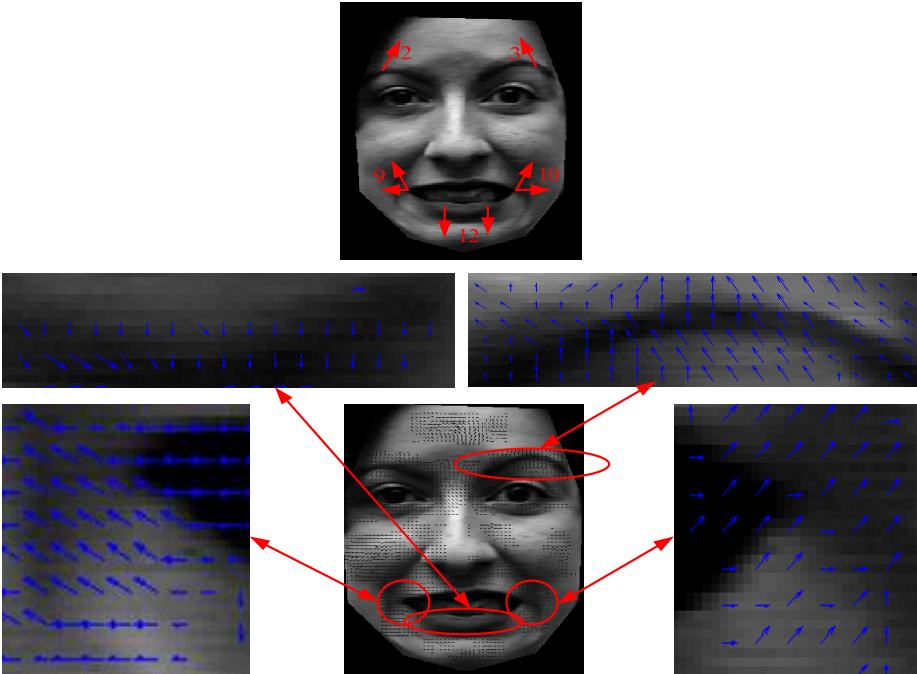


Figure 5.17: Vector Flows in ROI for facial expression of fear.

Happiness—mouth, brows lids:

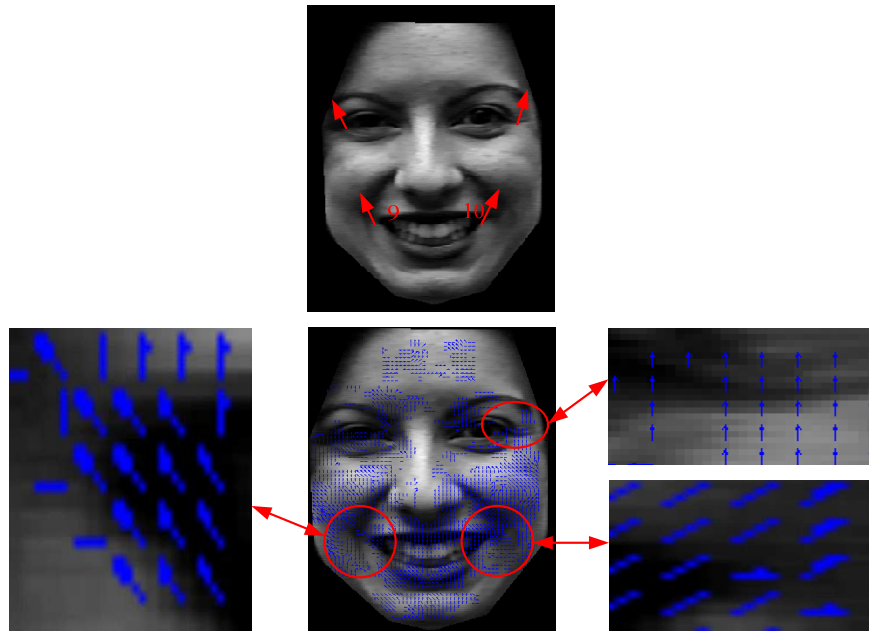


Figure 5.18: Vector Flows in ROI for facial expression of happiness.

Sadness—lower-lids, chin and lip corners:

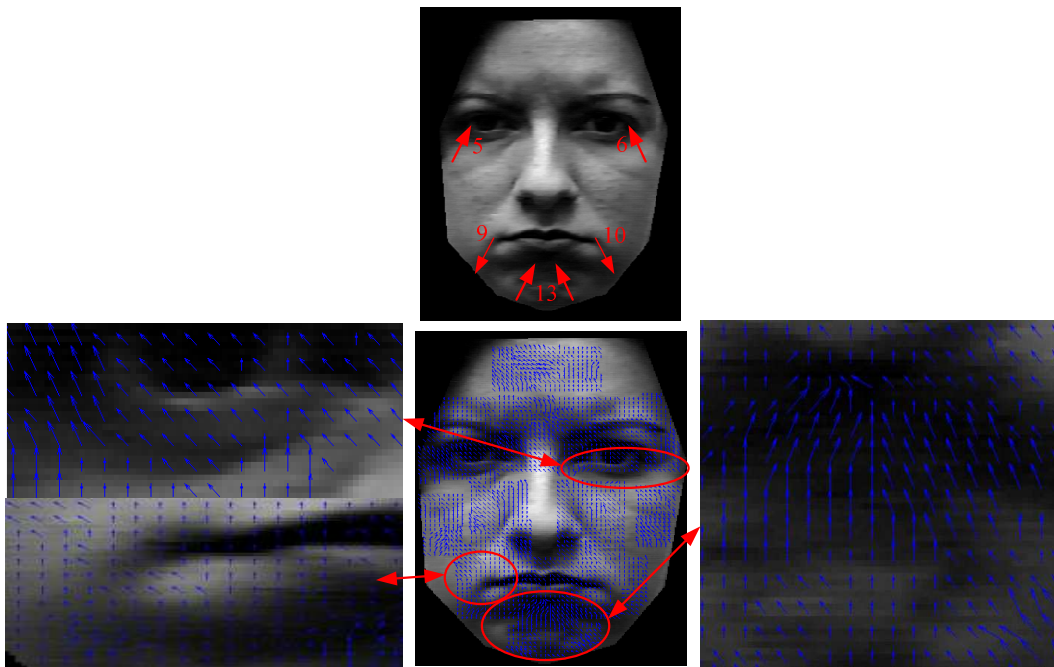


Figure 5.19: Vector Flows in ROI for facial expression of sadness.

Surprise—upper-eye lids and brows, mouth:

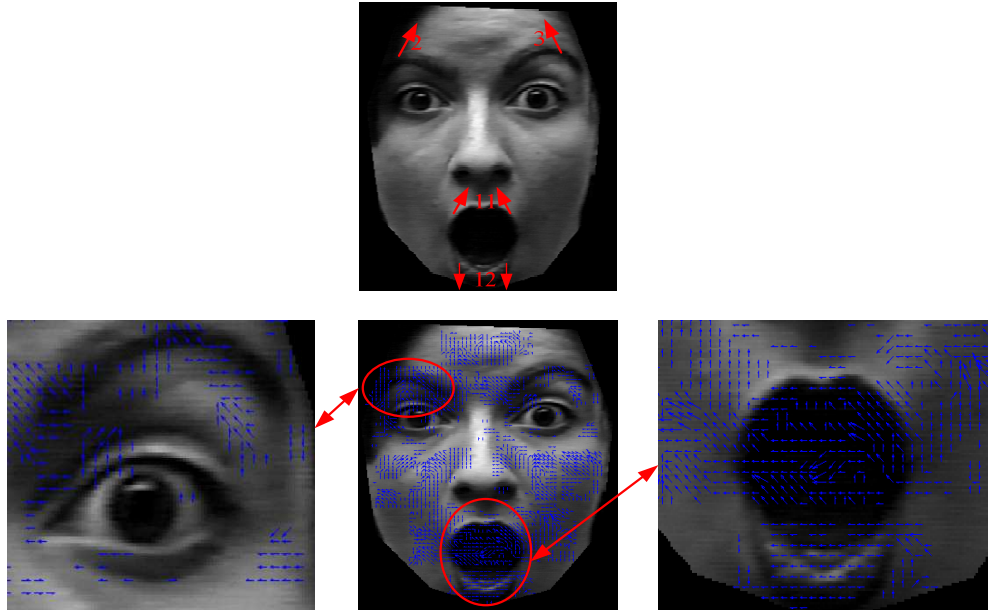


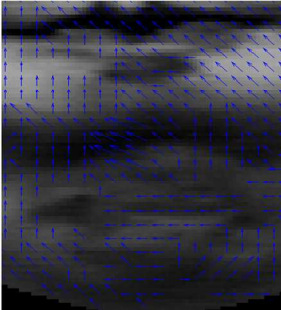
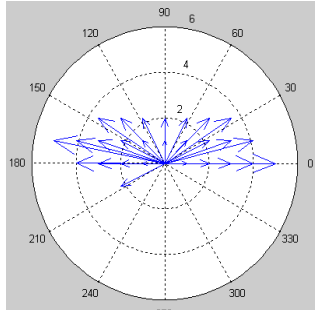
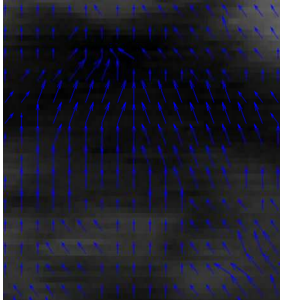
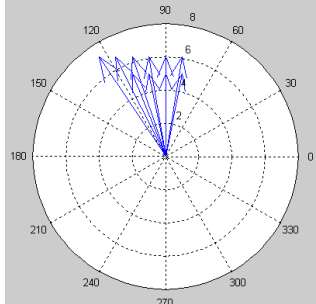

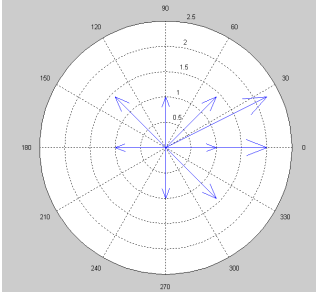
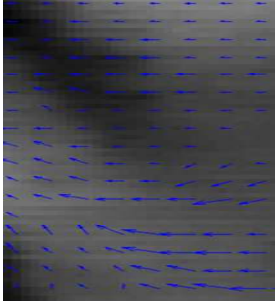
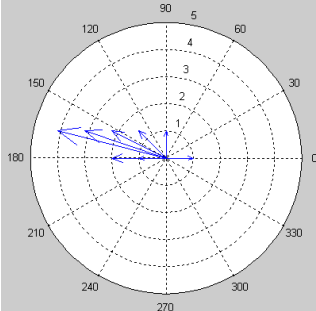
Figure 5.20: Vector Flows in ROI for facial expression of surprise.

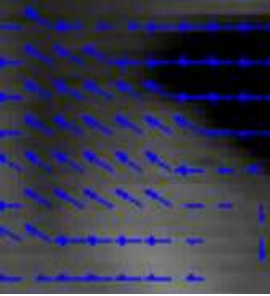
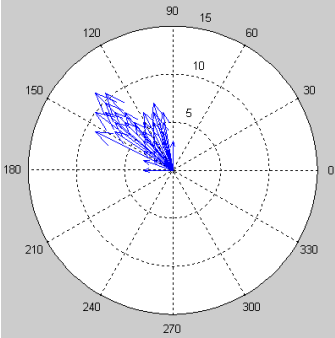

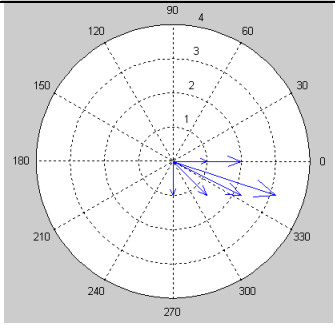

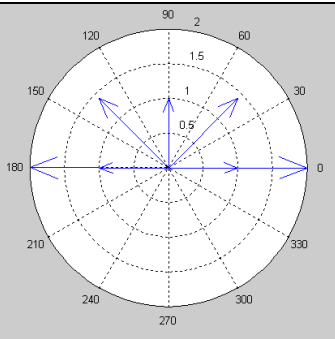
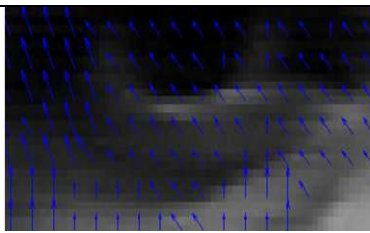
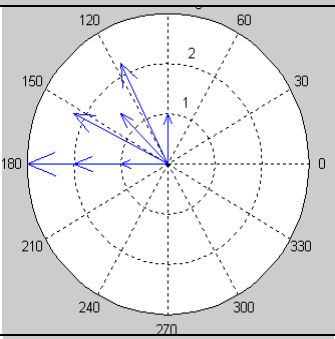
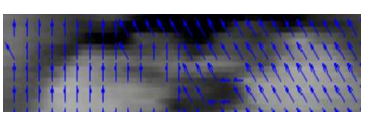
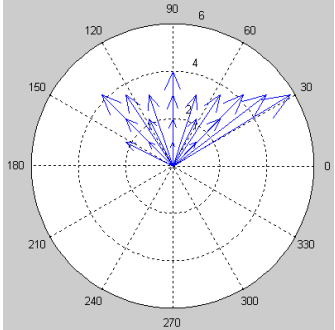
After given the vector flows shown in ROI for six basic emotions separately we used the same approach as in the analysis optical flows section to map those optical flows. Because in the experimental part we used direction and intensity of vector flows as the parameters now we should clearly map those vector flows and estimate their direction and intensity.

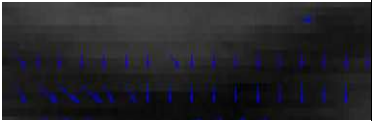
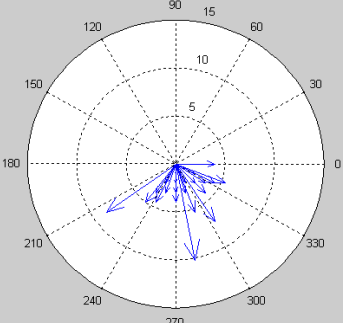
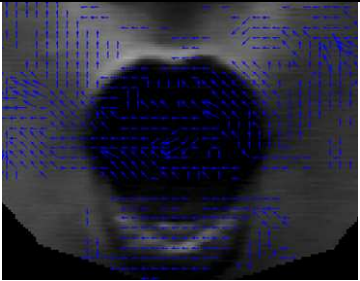
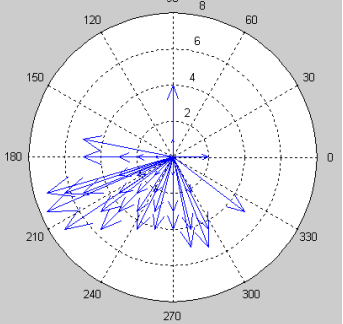
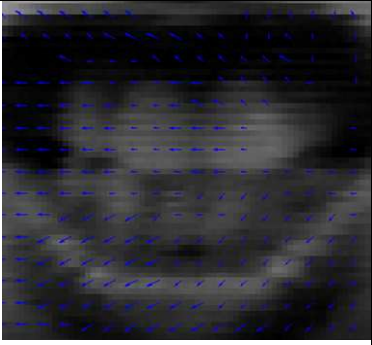
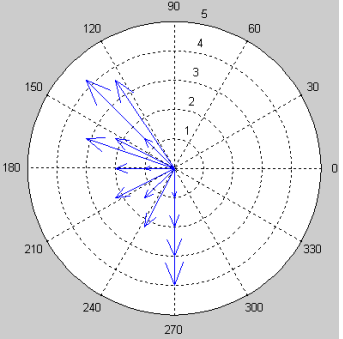
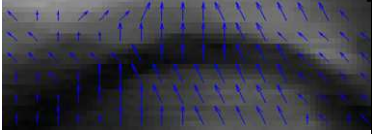
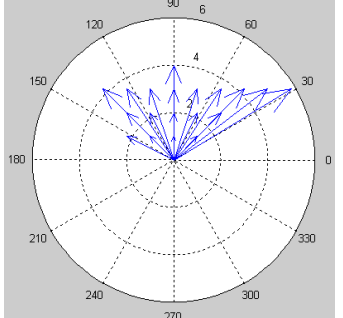
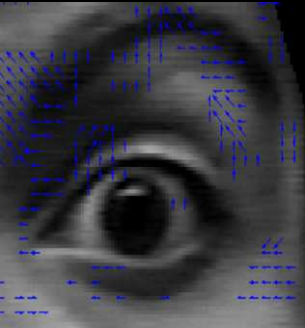
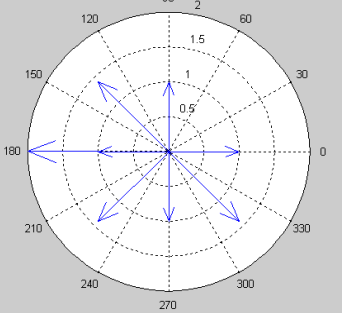
Optical flow analysis on each of the ROI images for six basic emotions (comprised with the baseline image which is “natural” expression) resulted in vector compass maps, some examples of which are shown in following Table 5.1.

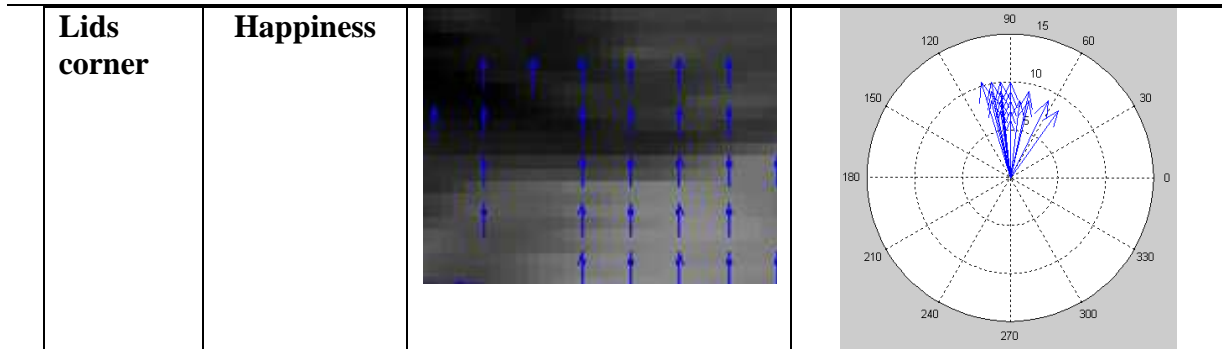
From above vector flow compass diagrams shown we can conclude that we chosen parameter average speed and direction are good representatives of the flow in the ROI. We notice that the vector flow movements shown in ROI are obvious different among 6 basic emotions. In addition, for example in very ROI there are no cycles of opposing vector flows. Because in that case the average parameters could be zero but the vector flow is unequal to zero other. Moreover, we see that it is not necessary to consider more regions of interest because all the vector flow movements are captured by the defined ROI. For example we don't need a ROI around the pupil of the eye, because the movement of the eye is captured by the ROI around the upper and lower eyelid. The gaze of direction is not important for detection of emotions. It is only important if the eye is open (surprise/fear) or closed/squeezed (happiness) or upper lid is falling down (sadness). So the set of ROI is a complete and unambiguous description of the six basic emotions.

Table 5.1: Table Optical flow analysis on each of the ROI images for six basic emotions.

ROI	SIX BASIC EMOTION	VECTOR FLOWS SHOWN	COMPASS DIAGRAMS (VECTORS MAP)
Chin	Anger		
	Sadness		
Wrinkle	Fear		
Wrinkle (nose) (right)	Disgust		

Lip Corner	Happiness (left)		
	Sadness (right)		
Lower lid	Anger		
	Sadness		
Lower lip	Anger		

	Fear		
Mouth	Surprise		
	Disgust		
Brow	Fear		
	Surprise		



This section is as an introduction of our model/ROI. In the next chapter, we will computer the activities of AU's using our model and DBN.

Chapter 6

Facial Expression Model and Implementation in SMILE&GeNIe

As basic reasoning/classification model DBN has been chosen as a tool that we use the DBN implementation in smile & GeNIe. To start we display a BN model for every basic emotion (GeNIe graphs). The ROI are nodes in the lowest level of the BN model. As discussed before every emotion can be described by the activation of one or more AUs. And for each ROI there are one or more related AUs. According to the different movements of vector flow the different AUs can be recognized in each ROI. Every basic emotion is characterized by a typical set of AUs. In appendix table we define 18 important AUs which are related to the basic emotion and which are the AUs we recognized.

Once we have our BN models we have to describe the probable relations between the variables in the CPT tables. In the last chapter we gave some examples of CPT tables. The last step is to fill the CPT tables. This is realized by training a BN on available data. So we also designed a dataset with the parameters computed and the BN model. In chapter Tools and Methods we described the working of the training algorithm and more details about modeling and CPT design will be given in following sections.

Finally we represent our model of DBN. We sample different points along the time axis and in every point $t = t_n$ we consider a time slice containing a BN. Next the nodes in different time slices are connected.

6.1 Overview

In this section we provide an overview of the manner in which the Dynamic Bayesian network has been constructed. During this process, we will discuss the thesis' guidelines, as well as our approach to develop a satisfying solution.

An approach was presented that uses a dynamic Bayesian network to model the relationships among different AUs [28]. In their approach nodes describe the observed features (AUs) of the system known to be relevant to the goals (emotion states) of the system optimization. In this thesis a multi-layered approach was used. We designed three layers in the DBN structure in which each layer has features. Then we model how the simple features combine into more complex ones.

6.2 The Multi-Layered Approach

In order to manipulate abstract entities like our representation of recognizing emotion from features extracted by the optical flow algorithm, we propose to decompose the problem into several layers. The highest level layers are the most abstract and specify “what the emotion is”, the intermediate level layers describe “the region of interest and related AUs”, and the low level layers present “how to recognize the AUs by optical flows by parameters estimation in those related regions”. Each level is seen as an independent layer with its own input and output. The advantages of the approach will be given following:

- The model is extensible. It is easier to implement and to test.
- The independence of each layer allows the behavior of a value of the system to be modified without impact on the others.

6.3 Module Design

This section provides the details of the global design of the constructed Dynamic Bayesian network. The following will discuss the implementation of lower-level nodes, the handling of higher-level features and finally the general structure of the network.

In order to develop a properly structured network, we propose a technique that relies on providing a set of low-level features for the activity present in a sequence of video frame, as opposed to the labeling of relevant features in the video. Given a trajectory of feature measurements and a label for a high-level event, our system first clusters the features to discover the lower-level events. We suppose that a high-level event is defined by a sequence of low-level events. In a given high level event, we track the object to do what we defined from one low-level features cluster to another. Then the probability of a given trajectory through low-level events will be given a particular high level event to determine the high level activity. So lower-level nodes must be provided that would serve as variables representing conclusions drawn from previous nodes' values.

After utilized lower-level nodes, the concept of higher-level features is revisited, coupled with their implementation in the Dynamic Bayesian network finally, we use each feature in the training data (dataset) to learn the relevant parameters, i.e. transition probabilities and observation likelihoods. These variables are discrete, so these can be represented as a table (CPT), which lists the probability that the child node takes on each of its different values for each combination of values of its parents.

Analog to the manner in which the DBN has been described above activates higher-level features based on their lower-level parts; the Dynamic Bayesian network should be capable of

deterministically enabling higher-level features when necessary. Region networks are a kind of network commonly used to structure more general kinds of information. Generally speaking, region networks are responsible for structuring low-level features. In our project, instead of relying on a traditional network node, higher-level features are translated to their Bayesian equivalent using deterministic network nodes. Instead of having probabilities which dictate the value of such nodes; their outcomes are precisely defined for every possible set of values of previous non-terminal nodes. As an example, this would effectively allow the network to activate higher-level feature C, if lower-level features A and B are activated.

After highlighting the approach in which higher-level features are handled, the general structure of the constructed Bayesian network is presented. When deciding on the arrangement of the network, it was of primary importance to reduce the number of incoming connections for each node, in order to prevent the conditional probability tables from growing too large.

Keeping the above in mind, in region layer each node contains merely 2 nodes (angle and length) connected to it. In region layer all nodes represent a measurement of the vector flows. The region level has a role in indicating estimation of parameters determined from the dataset input cues. Additionally, it can be observed that the incoming connections to the region nodes are not further defined. The illustrated nodes “lower level region processing” and “lower level AUs processing” represent networks consisting of lower-level feature nodes, higher-level feature nodes and non-terminals as previously discussed. Their specific formation was determined in the implementation phase of the thesis and will therefore be revealed later.

Furthermore, we use the contents of the conditional probability tables to deal with many different sources of uncertainty in problem solving, and the probabilities of the network’s nodes’ outcomes occurring based on the values of the preceding non-terminates, were determined in the implementation phase as well.

Considering the above issues to a discussion of the actual implementation of the network in section 3.2, this section has discussed the usage of non-terminates, the handling of higher-level features and the general structure of the network. Therewith, this chapter has been concluded. The following chapter outlines the implementation details of the developed dynamic Bayesian network, and finish CPTs.

6.4 Bayesian Network Modeling

In this part, we discuss the subsystems that form the research architecture and the sub-models in DBN structure. In the beginning we present a short overview of these subsystems by a flow chart and have a first view of how they relate to each other. Although creating models for temporal reasoning with DBNs sounds promising at first sight, many problems have been faced when we model it and a lot of questions remain to be answered. For instance: how do we create the structure and what parameters we should use to learn the structure? What is the performance of inference? How much data do we need for learning the parameters and which database is best for our system? Firstly, according to the relations between the facial expressions and facial AUs, in addition the relations between the interested regions and facial AUs we create a BN model to

represent the causal relations. Then we extend the BN to a DBN model for modeling the dynamic behaviors of facial expressions in every image frame.

The design of the BN causal structure should best reflect experts' understanding of the domain. According to the motion measurement by computing optical flow and the relations between AUs that are related to the six basic emotion expression categories in introduction part, we build the BN model as shown in Figure 6.1, which best represents the relationship between AUs and optical flows movements in each ROI for static face images. Our BN model of facial expression consists of three primary layers, namely, emotion expression classification layer contains 6 basic emotions, optical flow data layer contains ROI, and facial AU layer.

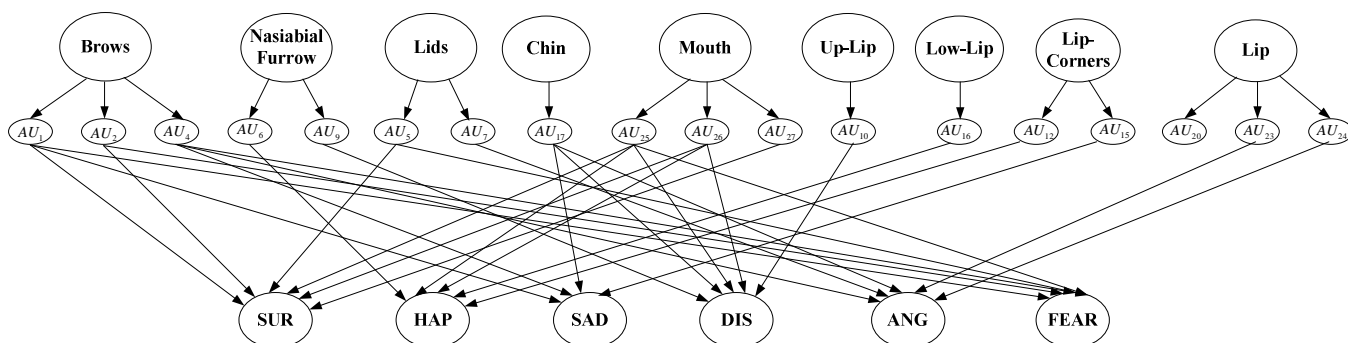


Figure 6.1: The BN model of six basic emotional expressions.

Note; HAU6 means AU6 belonging to happiness. Other notations in the Figure follow the same convention above.

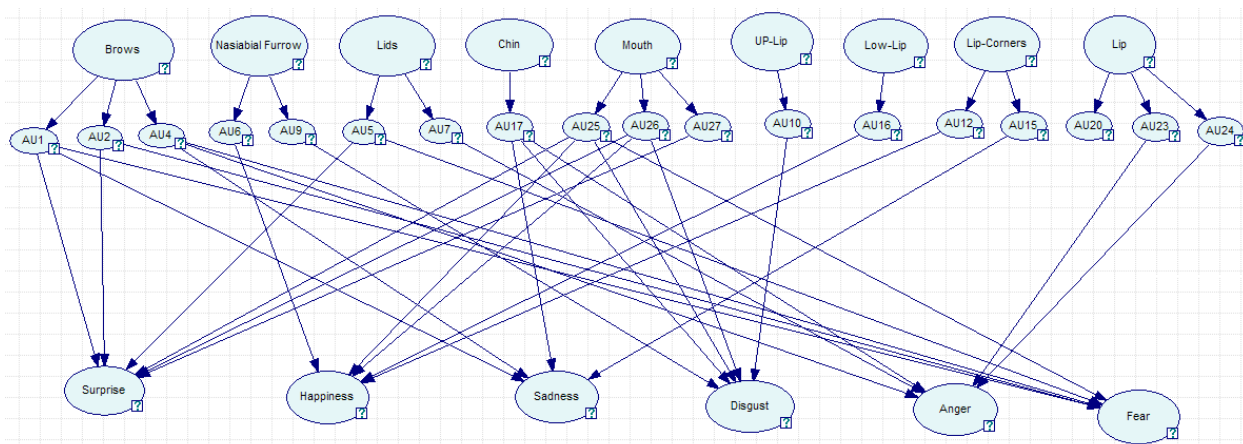


Figure 6.2: A BN model in GeNIe & SMILE.

The classification layer consists of a class (hypothesis) variable E including six states e_1, e_2, e_3, e_4, e_6 , which represent six basic emotion (happiness, sadness, disgust, surprise, anger, and fear) respectively, and a set of attribute variables denoted as HAP, ANG, SAD, DIS, SUR, and FEA corresponding to the six facial expressions as shown in Figure 6.2. The goal of this level of abstraction is to find the probability of class state e_i , which represents the chance of

class state e_i given facial observations. When this probability is maximal, it has the largest chance that the observed facial expression belongs to the state of class variable e_i .

The optical flow level of layers in the model is the data layer containing optical flow information variables, which are the lowest level of layers in our model. We defined nine regions of interests in which the optical flows have been detected by Lucas and Kanade algorithm. The nine ROI are Brows, Lips, Lip Corners, Eyelids, Cheeks, Chin, Mouth, Nasolabial Furrow, and Wrinkles, which contain the optical flow information variables which are $\overline{V}_x, \overline{V}_y$ and average angle for each region. But for the six basic emotions we only choose the most related regions among the 9 ROI which have the most prominent change. We did not consider all of the nine ROI for every emotion because we considered the limits of nodes and the complexity of our model. For each ROI the related AUs were given in the higher level. For example, for Brows region the related AUs are AU1 (Raised inner brows), AU2 (Raised outer brow) and AU4 (Lowered brows or Frowned brows). The details of defining the ROI in facial expression and choosing the related ROI for every emotion have been presented in appendix table. All variables in this layer are observable. In our implementation, for facial features in pair such as brows, eyelids, and eye wrinkle, and Nasolabial fold, we measure the optical flow change on both sides of the face and consider the one with the prominent change as evidence.

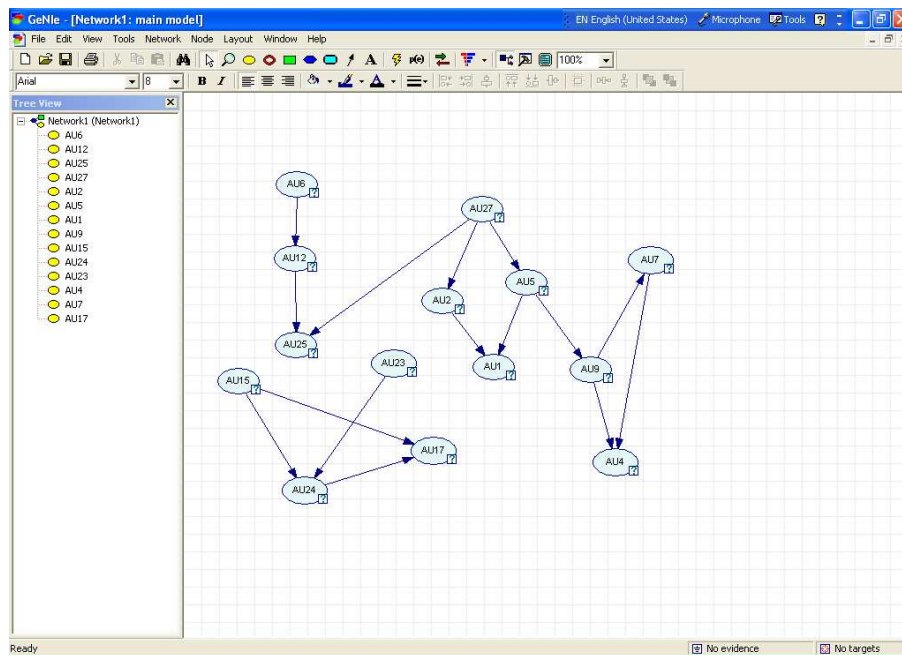


Figure 6.3: The model of the probabilities of each possible pair of features co-occurring.

Because co-occurrence exists among AUs, it is really difficult and not reliably to recognize AUs individually. But fortunately, as described in the FACS manual there are some inherent relationships among AUs. According to the relationships among AUs we know the related regions for each AU. For example, the co-occurrence between AU12 and AU16 is very high because both them are related motion of lips. The following model (Figure. 6.3) shows the probabilities of each possible pair of co-occurring features.

Considered the co-occurrence mentioned above. In our model for those most related ROI we also separate them to primary regions and auxiliary regions. A primary region contributes the most prominent difference among another emotion expression. For example, for both sadness and fear expression we can see the changing of optical flow in lip corners but the difference between the two shapes is very obviously. The auxiliary ROI means the changing of optical flow in those regions exists but it is not so obviously and the changes are usually connected to the primary ROIs' changing. Hence, the likelihood of primary ROI to the facial expression is higher than those of auxiliary ROI.

The label of the layer above the AU layer is analogous to linguistic description of the relation between AUs and facial expressions. And for every emotion expression in the related ROI the AUs that are related the six facial expressions have been given.

In the last level, we will use an expert system by programming in CLIPs to extract the emotions instead of using the same approach to extract emotions from related AUs since there is a problem of the Cohn Kanda database. The video only provides the list of activated and deactivated AU's but doesn't provide labels for emotions. In fact some videos do not show any emotion but is just to show the impact of one or more AU's. Many researches were using a subset and attach some labels to the database, but we have our doubts about the validity of the labels. But the labels with AUs in the database are correct and done by Cohn himself. So we have to say the database is only suitable for recognizing AUs instead of emotions.

6.5 Dynamic Bayesian Network Modeling

Facial expressions can be seen as a good way to express emotions and show the feeling during the communication. Emotions always vary according to the interaction environment so the duration of facial expression is often varied related to the emotion underlying the expressions. We can safely say that the facial expressions in duration of communication is a temporal behavior, that is to say, in different time slices the behavior is different.

Modeling such temporal behaviors, the static BN model is not good enough because static BN works with evidences and beliefs only from a single time instance, that is to say, it is unable to express temporal behaviors because some temporal dependencies are included in the consecutive occurrences of expression in frame sequences. According to the limitation, we propose DBNs to model the dynamic duration of a facial expression. Based on the BN which represents the relationship between AUs and optical flows for each related regions for static face images we use DBNs for modeling the aspect of interconnected time slices during a facial expression. In the highest level of DBN the interconnected time slices are presented, and in the DBN model we use the first order HMM to model the relationships between two neighbors.

6.6 Model Setup in GeNIe & SMILE

In this section we present the implementation details underlying the developed solutions. First, we will construct dynamic Bayesian network to distinguish between AUs and some activated features that have been defined in previous sections. Thereafter, we plan to discuss the intricate details of the Bayesian network's implementation in SMILE and the details of building graphical decision models and performing diagnosis in GeNIe.

This paragraph outlines the implementation aspects of the network. Previously we described it from a conceptual point of view. In the remainder of this section we will provide the intricate details of the development model's internal workings. We will give a closer examination of the rules and dataset are founded on and expose the lower-level and higher-level details of the application.

- **Implementation details**

In this paragraph we proceed to describe the implementation details through analyzing the DBN's procedure. The procedure can be performed by following separate steps which is regarded in a sequential fashion, highlighting their underlying details.

- **Dynamic Bayesian network**

A Bayesian network is a high-level representation of a probability distribution over a set of variables that are used for building a model of the problem domain. In this paragraph we aim to describe the implementation details of the constructed Dynamic Bayesian network. In order to achieve this, the following we will start off with presenting and discussing the network's final structural organization. Next, we will provide the technical details, primarily focusing on the Bayesian network's conditional probability tables (CPT).

- **Optical flow-region level**

Having discussed the structure of the AUs-emotion states level part of the network, the following regards the determination of the AUs based on region cues. The optical flow-region level length/angle of the Bayesian network is slightly more complex compared to AUs-emotion level length/angle because a lot of researches have already done on recognizing facial expression by AUs which we have already mentioned in chapter 2. The value of the optical flow-region level node is based on the input it receives from further subdivisions of the network. Each of these networks has a structural setup similar to the AUs-emotion level length/angle of the network. Besides their unique contents, one more difference can be distinguished, as described in the following.

Level 1: First create a static BN containing nodes: Angle, Length and Region and two arcs going separately from Angle to Region and from Length to Region, just as what we create it in a normal way. Each region contains three parameters which are length and angle (V_x, V_y, A) .

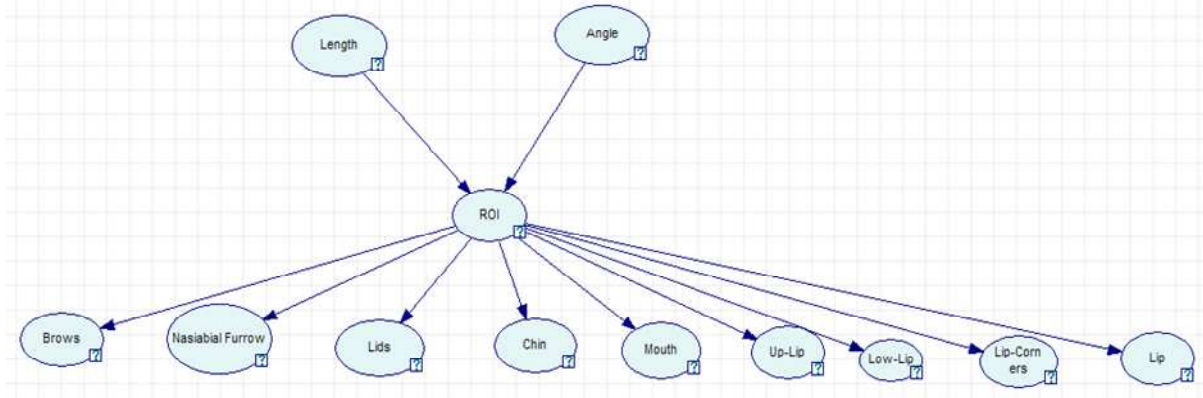


Figure 6.4: A BN model Level 1 in GeNIe&SMILE.

Region-AUs level

As previously described at the beginning of this paragraph, one of the sub models is the Region-AUs level. The mask has been divided into 9 ROI and for each AU we defined the related regions which means for each AU these regions may be changed.

Level 2: First create a static BN containing nodes: Regions, Aus and arcs nodes going from Regions to AUs.

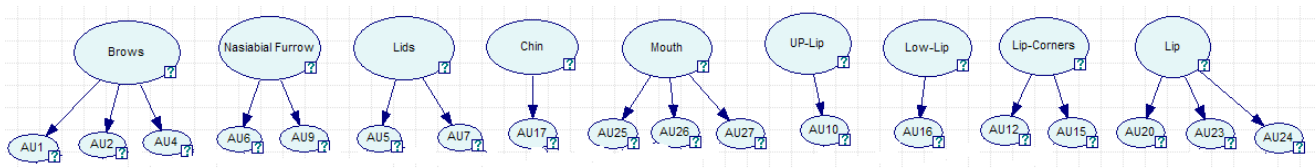


Figure 6.5: A BN model in Level 2 in GeNIe&SMILE.

AUs –Emotion States level

As previously described in chapter 4, our network’s final output node are some AUs related to six basic emotion states, providing an indication of the general AUs-level of the observed situation, as represented by the set of activated features. Figure 6.1 illustrated the global structure of the network, separating the features to 3 levels. Following we will reveal the manner in which the feature nodes at the network’s roots are connected to obtain these higher-levels that is not defined previously. The final layout of the Bayesian network can be observed in Figure 6.6. The emotion state level is solely determined from the input of several AUs nodes which are combinations of a number of lower-level feature nodes. Through means of the intermediate nodes, it has been attempted to effectively group the raw feature nodes into a single network element. However, the intermediate nodes’ results should be regarded as an interpretation of their preceding network elements. These interpretations are mandatory in order to finally obtain a global measure in the form of a single node’s output.

Level 3: First create a static BN containing nodes: AUs and Emotion States and arcs nodes going from AUs to Emotion States, just as we would do normally.

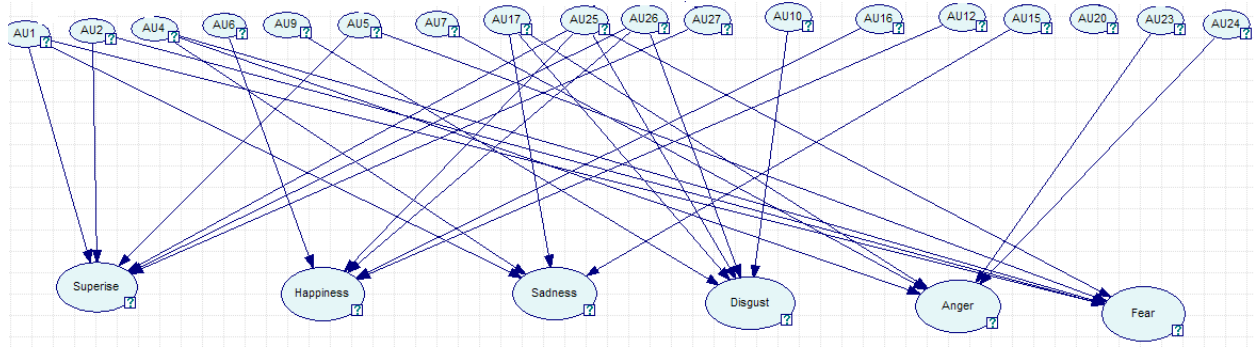


Figure 6.6: A BN model in Level 3 in GeNIe&SMILE.

For this level we use an expert system which is a convenient way to determine the emotion from several image sequences which represent six basic emotions. The results are good; the implementation is straightforward so we only need to Figure out the best rules.

The program gives the result that is expected from the rules. The results also depend on the threshold values which would be further researched in the future.

6.7 Software Design

The research goal of the thesis was to obtain an initial insight into the applicability and usability of the BN module in modeling the processes of recognizing AUs. In previous part of this thesis, we showed the facial expression model, how to extract the ROI and how to analyze vector flows in each ROI. In this chapter, a research is performed to answer our research questions presented in the beginning of the thesis: how do we obtain the structure of module and parameters? How to train the model? What is the performance of inference? How to design the dataset to training and how much data do we need for learning the parameters? So in this chapter, we will present more details about implementation in GeNIe and SMILE software. The main implementation in GeNIe and SMILE software is how to create the DBN structure and the learning of the parameters. Special-propose tools were programmed in C++ and Matlab for processing images, data discretization, parameter learning, and the inference study.

A general overview of the steps for modeling processes of recognizing AUs with DBN is given in Figure 7.1, including some research questions and considerations regarding these steps. This Figure is derived from the global research architecture presented in chapter 4. In Figure 6.7 we presented more details about modeling, parameters learning and inference.

6.7.1 Parameters Estimation

Parameters estimation is implemented in Matlab. Firstly, we use Lucas-Kanade method to implement optical flow algorithm in Matlab. Secondly, we implement contour and corner algorithm to detect ROI. Finally, we compute the average angle and length of vector flows in each ROI which are the parameters we used to learning in model. All vector flows can be obtained from Lucas-Kanade method. But it is impossible to use those data to learn the DBN parameters. Before use those data to learn parameters in DBN model it needs to be processed. In this particular case, each image is divided into 7 frames and sampling is fixed to 2 seconds between each sample. But for showing the significant difference among different emotions we only choose the starting frame and the last frame to compute parameters.

During the different steps of experiments the special-purpose software should be written to implement. The main steps existed in every experiment are almost the same: 1. Extracting feature in Matlab using Lucas-Kanade method, 2. Discretizing and sampling the data obtained, 3. Parameters estimation according to extraction features, 4. Learning the BN parameters and performing inference with c++/SMILE.

The detail of the code has been presented in the appendix. Here we just use a diagram to give a general overview of the classes and data types appeared in the program. And the interaction between the classes is also presented. There are not too many implementation details which can be given, because SMILE is closed-source and we only use some classes in SMILE library to implement our work.

Chapter 7

Experimental Results

We developed models, designed and improved algorithms and set up our system for automatic recognition of AUs in chapter 5 and chapter 6. In this chapter we will test our system. So we first present the test-database and next the performance evaluation will be focused on localization of ROI and AUs recognition.

Firstly, Section 7.1 provides a brief description of the database used to test our approach. In section 7.2 the results of the landmark localization were evaluated by the proposed rectangular measure that represented a localization result as a rectangular box. Then section 7.3 and 7.4 gives experimental results which contain the result of recognizing AUs and recognizing 6 basic emotions, moreover the comparisons were given. In order to validate the obtained results of our approach, we compare it with classification done by human observers, and with results obtained in previous related work.

7.1 Cohn-Kanade Dataset

To train our system we used the Cohn-Kanade database which consists of video recordings of facial expressions, starting from a neutral expression and ending most of the time in the peak of the facial expression. Subjects are university students enrolled in introductory psychology classes. They ranged in age from 18 to 30 years. Subjects were instructed by an experimenter to perform a series of 23 facial displays. Six of the displays were based on descriptions of prototypic emotions (i.e., happiness, anger, fear, disgust, sadness and surprise). There are 104 subjects in the database and only 10 of them gave the consent for publications.



Figure 7.1: Extracting the prominent facial region and resizing to 32 x 32 bit size.

It is apparent that the ground truth for each image is taken from the emotion that the subject "claims" to be expressing, and not by what human observers "believe" he/she is feeling. This is a critical point, and could be considered as a shortcoming of the data-set. Subjects are only "acting" and are not sincerely "feeling" a given emotion. In the future work one of the most challenging topics the creation of a natural database of pure emotions

Each image in the database is in RGB bitmap format with a size of 320 x 240 pixels. For the purposes of our work in this thesis, we were interested in extracting the best frontal views to characterize each class of emotional facial expressions. Each expression class is shown starting from a neutral expression and ending most of the time in the apex of the facial expression, but we only detect features from the apex of the facial expression because in the apex the movements of optical flows are most intense. We selected from the data set, 33 surprise, 26 happy and 33 angry 21 fear, 20 sad and 23 disgust images which contain those important AUs we want to recognize. Next we complete the total to 135 training/validation images which are used by our learning algorithm. We pre-extracted the main facial regions of each image using the same technique for automatic face localization described in chapter 5.

7.2 Localization of Facial Landmarks for AUs Recognition

A fully automatic method was designed for feature-based localization of facial landmarks in static grey-scale images. One important problem in earlier method was that the conjunction of AUs or AU combinations caused a wrong localization error in the landmark localization. After applying a procedure of x/y-contour projection in many cases some merged regions were successfully separated. In particular, the merged nose-and-mouth and eye-and-eyebrows was successfully separated. The algorithm and improvements have been presented in chapter 5. In this section the performance of the method will be evaluated.

7.2.1 Performance Evaluation Measure

The main improvements to the existing version of the method are the local contrast threshold calculated in every filter neighborhood. In previous method, in order to define a threshold for contrast filtering of the located contour regions only a double average contrast of the whole image was used. In the earlier version of the method because of erroneous connection of contours belonging to different facial landmarks into one region, two or more regions were probably merged. But for our objective to recognize AUs it is necessary to separate regions in a face. For example, AUs 10 (upper lip raiser), 11 (nasolabial furrow deeper), and 12 (lip corner puller).

To separate the neighboring candidates merged into one region we proposed a simple but effective technique of x/y-contour projection. The details of the technique have been presented in chapter 5, here we only show the merged facial landmarks and their separation by x/y-contour projection. The contour points were projected to x-axis for upper face landmarks and to y-axis for lower face landmarks. If a landmark candidate consists of two or more regions of contour concentration. The x projections mean a number of contour points along the corresponding columns or rows of the contour map for the upper face landmarks. The y projections mean a number of contour points along the corresponding columns or rows of the contour map for the

lower face landmarks. The average contrast of the whole smoothed low resolution image was used to define a threshold (equation 7 in chapter 5). After getting the x/y projections, the number of projected contour points should be compared with a threshold. If the number of projected contour points was smaller than the threshold, contour points were eliminated. After several steps of contour elimination the initial threshold was changed to a minimum number of contours in the column or row of the given candidate.

In order to find a proper spatial arrangement of the located candidate, the proposed algorithm in chapter 5 applied a set of verification rules which were based on the face geometry model. The knowledge of face geometry taken from Farkas was very important to separate different regions in a face. The average distance between upper face landmarks both eyes and eyebrows and eyebrow-eye distance is significant useful for the purpose of structural correction of the located candidates. After all images were averaged all the characteristic points were manually annotated in the database. Further, bounding boxes were built on the basis of the selected characteristic points for each landmark in the database. The location and size of the bounding box were calculated from the coordinates of the top, left, right, and bottom boundaries of the contour region. The centre of the landmark was defined as the centre point of the bounding box.

7.2.2 Evaluation Results and Conclusion

A good performance measure to evaluate is the rectangular measure which verifies if is that the located landmark region is inside the bounding box length/angle or at least that inside the bounding box length/angle surrounding landmark was less than the actual size of the landmark. Figure 7.2 shows some examples of correctly localized facial landmarks in Cohn-kanade database. The following Figures are the facial expression of surprise, sadness, anger, disgust, fear and happiness separately.

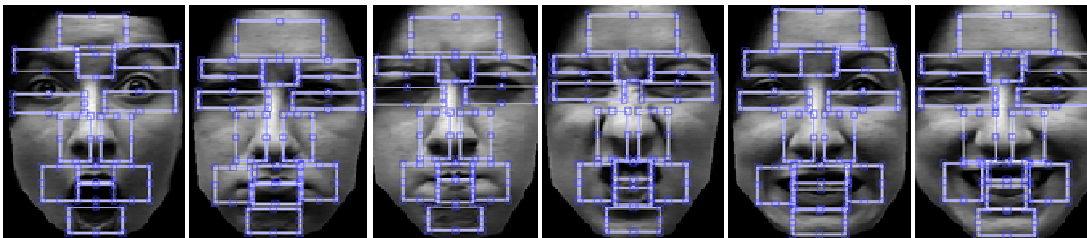


Figure 7.2 Examples of correctly localized facial landmarks of six basic emotions.

In upper face landmarks with wrong locations were mainly due to the classification of eyebrows and eye regions as eyes. But there are some AUs only related to eyebrows. In lower face the wrong locations were mainly due to the classification of nose and mouth or as one region. After testing 486 images in Cohn-Kanade database, as was expected, the wrong localizations occurred mainly due to the effect of lower face AUs 9, 10, 11 and 12 activated alone or in combinations with other AUs. Those AUs are activated in expressions of anger, disgust and happiness. AUs 4, 6 and 7 sometimes caused the merging of the upper face landmarks into one region. Some localization errors are illustrated as examples in Figure 7.3.

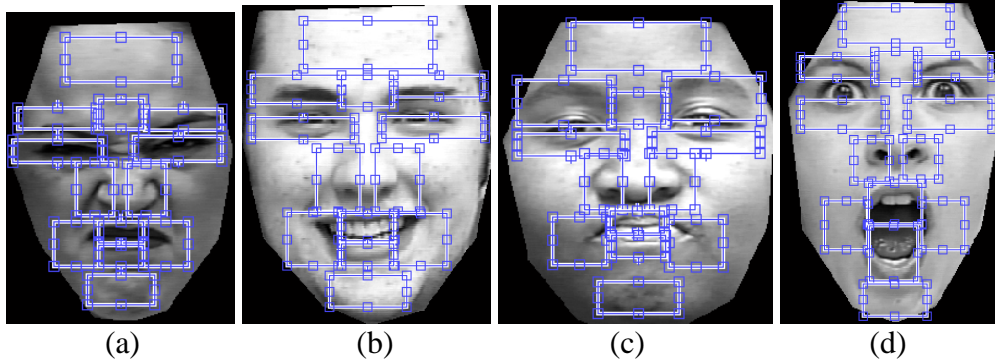


Figure 7.3. Examples of errors of the landmark localization: (a) , (c) eyebrows and eyes were misclassified as one region; (b) teeth was classified as lip; (d) mouth was merged with tongue.

Due to the fact that some AUs were not presented in the dataset or the number of images was too few, only a limited number of AUs and AU combinations was applied. As it is seen from the Tables 7.1 to 7.3, average localization rates of AUs and AU combinations in upper and lower face is above 80%. And the evaluations demonstrated a sufficiently high overall performance of the method (91%) that is comparable to some extent with performance of other point measure method.

Table 7.1: Improvement over upper face AUs and AU combinations.

Study /Aus	1	2	4	5	7	1+2	1+4	1+7	2+4	2+5	4+5	4+6	6+7
Ave Loc Rate, %	81	84	88	86	83	84	82	89	80	82	84	85	77

Table 7.2: Improvement over lower face AUs.

Study /Aus	9	10	11	12	15	16	17	20	23	24	25	26	27
Ave Loc Rate, %	84	82	88	76	89	89	87	81	88	87	85	89	84

Table 7.3: Improvement over lower face AU combinations.

Study /Aus	9+17	11+20	11+25	12+25	16+25	17+23	17+24	23+24	25+26	11+25	25+27
Ave Loc Rate, %	81	87	81	85	83	77	78	90	88	89	86

7.3 AUs and AU Combinations Recognition

Having mentioned the method of extracting features and defined the parameters that will be used to estimate vector flows, in the following section we will give more details about how to compute those parameters and how to prepare the dataset from the Cohn-Kanade database for training.

7.3.1 Training Dataset Design

Remember that the goal of our research is to develop a new recognition method of facial actions based on the features detected from vector flows. Our aim is a more accurately capture of the characteristics of facial motion instead of just employ AUs to recognize facial motion.

Considering that the dataset should be designed as a lot of numbers in a matrix as a text form we have to prepare vector flows and use parameters to extract. Moreover, because the first goal of our research is to use vector flow to recognize AUs one of the first steps is to find the relations between vector flows and AUs.

To start we have to recognize the region where an AU is most active. For two different frames we can compute vector flows and according to the intensity of vectors we can know which parts in the face are moved. Maybe there are some parts which are moved a little and they are not the most important part that we want to consider. For finding the main motion region we have to define a minimal motion energy $M_{min} = 0.01$. This is the motion threshold, which is the sum of vector lengths across all flow regions and the average angle of vector flows. If M is less than M_{min} facial motion was too small for recognition. Moreover we have to present the regions which contain maximal motion. When we choose parameters to be trained we consider only those regions which have the motion energy that is higher than the minimal motion energy because the motion energy is determined by the intensity of vectors flow.

7.3.2 Features Weighted

The initially incoming activated features are all low-level and have to be examined in order to activate the appropriate high-level features, if any. In our model the parameters of vector flows are low-level features which activate the AUs which are viewed as high-level features.

Since high-level features may activate further features, the process is repeated until no additional features are engaged. When the dataset is trained and all the procedure is completed in DBN, the final set of activated features is available to compute CPTs in different layers.

Once the final set of activated features is obtained, the DBN proceeds to make an initial recognition according to CPTs. The CPTs represent the scenarios that are potentially occurring, based on the available feature data.

Having obtained the annotated recordings that are possibly occurring the application attempts to verify its recognition through means of a search for confirming or supporting features which have high probabilities according to CPTs. Besides its main features, each emotion state contains a set of pre-defined supporting features, that is to say, we may choose some features with the highest probabilities.

For each supporting feature that is currently active, the emotion state that is under consideration is more likely to be selected as the final dominant hypothesis. Since not all supporting features are equally important which means all supporting features may have different weights, their activations are weighed according to a set of CPTs by training the dataset in BN that we designed.

The CPT tables are constructed for six basic emotions. Each table presents one emotion with related AUs and the related ROI for each AU. Each ROI contains three variables which are Length and Angle.

In our framework, we designed four level DBN based learning schemes and the four layers have already been described in previous section. The training data consists of the angle and length of vector flows labeled from Cohn-Kanade for each frame in each region.

In the training model, the random variables length /angle denote the difference between two vectors and angle of vector flows in each region, respectively. We take the difference feature from the neutral and apex frame. The hidden variables are ROI. According to the motion of vector flows in each region we got the motion of this region. As we have defined the related regions for each AUs and have defined the motion of those related regions for presenting each AU as well. In the following section we present how to get CPTs for the 18 AUs that we want to recognize related to six basic emotions separately.

7.3.3 Experimental in GeNIe & SMILE Setup

In section 7.1 we described Cohn-Kanade Database. In this section we present more details about training data and our preliminary testing results.

7.3.3.1 Training

The primary goal in this thesis is to show how our approach gives reliable and robust expression recognition using our feature extraction and facial region localization technique modeling by BN. To this end, GeNIe was trained to be able to classify any of the images in Cohn-Kanade database. First we built a data set, labeled according to the parameters estimated from our feature extraction and automatically facial region localization system. Then we decided to use EM algorithm to train the dataset.

The procedure for learning parameters in GeNIe is as follows:

1. Create a data file where each column maps to a variable in the network. The top row in the data file should contain the variable names.
2. Implement the network in GeNIe.
3. Load the data file in GeNIe.
4. Run the EM algorithm.
5. The output is our network in which the parameters have been learned.

After training the dataset in GeNIe we can update the permanent features. In the next step those parameters were used as testing data. The parameters used to represent the face feature for AU recognition are presented in the Appendix.

7.3.3.2 Testing

The images we used to test are also from Cohn-Kanade database. The testing set contains 1140 selected images and testing is concerned with the automatic localization of image features. From the Cohn-Kanade database, we selected image sequences in which both a signal AU and AU combinations occurred in the face. Features were detected from all image frames but only the initial and final two frames in each image sequence were used for training or testing.

As shown in table 7.4, the image sequences were assigned to training and testing sets. In the first set S1, the sequences were randomly selected, so the same subjects maybe appear in both training and testing sets. But in the second set S2 no subject were allowed to appear in both training and testing sets, that is to say, the training data and testing data were totally different.

Table 7.4 Training and Testing Dataset.

Data Set		number of Sequences	Single AUs																		
			1	2	4	5	6	7	9	10	12	15	16	17	20	23	24	25	25	26	27
S1	Train	689	23	34	24	37	22	21	19	17	24	26	29	28	32	17	19	19	28	27	28
	Test	184	9	10	7	13	10	9	11	10	12	10	10	11	10	9	7	8	9	9	10
S2	Train	505	26	38	25	29	29	26	22	19	22	23	19	18	34	33	28	25	29	29	31
	Test	150	7	8	6	5	6	7	8	6	9	7	8	6	8	9	9	11	11	9	10

7.4 Experimental Results of AUs Recognition

We conducted three experiments to evaluate the performance of our system. The first is about recognizing single AUs when the image sequences contain only a single AU. The second is single AU experiments recognition for image sequences containing both single AUs and combinations. The third is about AU combinations recognition instead of only single AUs. Of course the image sequences should contain AU combinations. The experiment is also about recognizing signal AUs but in this experiment we recognize AUs in six basic facial expressions separately. The three experiments evaluate the approach of our model by using a database for training and testing.

7.4.1 Single AUs Recognition for Image Sequences Containing Single AUs

Table 7.5: Single AUs Recognition for Image Sequences Containing Single AUs.

Action Units	Recognition Rate																	
	1	2	4	5	6	7	9	10	12	15	16	17	20	23	24	25	26	27
AU 1	81.4	20.3	11.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AU 2	24.3	79.1	12.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AU 4	12.4	21.4	78.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AU 5	0	0	0	76.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AU 6	0	0	0	0	72.4	17.9	0	0	0	0	0	0	0	0	0	0	0	0
AU 7	0	0	0	0	14.6	69.6	0	0	0	0	0	0	0	0	0	0	0	0
AU 9	0	0	0	0	0	0	89.3	0	0	0	0	0	0	0	0	0	0	0
AU 10	0	0	0	0	0	0	0	65.3	12.1	9.5	6.4	0	0	0	0	0	0	0
AU 12	0	0	0	0	0	0	0	12.9	56.9	11.5	9.1	0	0	0	0	0	0	0
AU 15	0	0	0	0	0	0	0	5.1	11.2	68.1	21.4	0	0	0	0	0	0	0
AU 16	0	0	0	0	0	0	0	11.2	12.2	65.1	11.1	0	0	0	0	0	0	0
AU 17	0	0	0	0	0	0	0	0	0	0	0	84.1	0	0	0	0	0	0
AU 20	0	0	0	0	0	0	0	0	0	0	4.1	6.2	56.1	11.1	9.1	0	0	0
AU 23	0	0	0	0	0	0	0	0	0	0	4.2	2.6	12.4	51.2	11.4	0	0	0
AU 24	0	0	0	0	0	0	0	0	0	0	3.2	2.7	11.9	12.5	53.1	0	0	0
AU 25	0	0	0	0	0	0	0	0	0	0	0	0	9.1	2.1	5.6	57.1	13.4	5.1
AU 26	0	0	0	0	0	0	0	0	0	0	0	0	4.1	6.8	11.4	12.5	51.0	6.1
AU 27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13.1	14.6	68.1

In the first experiment, we used a BN model recognizer having the structure shown before to recognize only single AUs. The model structure represents the relationship between each ROI and related AUs. The inputs to the network were the face feature parameters shown in our model. The outputs of each sub-model were the related AUs which are parts of 18 important AUs:

(AU1,AU2,AU4,AU5,AU6,AU7,AU9,AU10,AU12,AU15,AU16,AU17,AU20,AU23,AU24,AU25,AU26,AU27). In addition, the output nodes showed the recognized AUs with the recognition rates. The recognized AU with the highest recognition rate was interpreted as the recognized AU.

7.4.2 Single AUs Recognition for Image Sequences Containing AUs Combinations

In the second experiment, we also used the same BN model as in the first experiment, and the inputs and outputs are also the same. The difference is that the images sequences were trained and tested which not only contain the single AUs but also contain AUs combinations.

Table 7.6: Single AUs Recognition for Image Sequences Containing AUs combinations

Action Units	Recognition Rate																	
	1	2	4	5	6	7	9	10	12	15	16	17	20	23	24	25	26	27
AU 1	89.4	20.3	13.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AU 2	29.3	79.8	11.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AU 4	12.4	21.4	88.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AU 5	0	0	0	86.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AU 6	0	0	0	0	81.9	14.9	0	0	0	0	0	0	0	0	0	0	0	0
AU 7	0	0	0	0	17.6	79.6	0	0	0	0	0	0	0	0	0	0	0	0
AU 9	0	0	0	0	0	0	79.9	0	0	0	0	0	0	0	0	0	0	0
AU 10	0	0	0	0	0	0	0	85.1	10.9	9.9	6.4	0	0	0	0	0	0	0
AU 12	0	0	0	0	0	0	0	16.9	79.9	14.5	8.1	0	0	0	0	0	0	0
AU 15	0	0	0	0	0	0	0	9.1	10.2	87.4	11.7	0	0	0	0	0	0	0
AU 16	0	0	0	0	0	0	0	7.2	9.2	92.1	3.1	0	0	0	0	0	0	0
AU 17	0	0	0	0	0	0	0	0	0	0	0	88.1	0	0	0	0	0	0
AU 20	0	0	0	0	0	0	0	0	0	0	5.1	7.2	85.1	6.1	9.8	0	0	0
AU 23	0	0	0	0	0	0	0	0	0	0	4.2	2.9	8.9	80.2	10.6	0	0	0
AU 24	0	0	0	0	0	0	0	0	0	0	5.2	4.1	9.9	10.9	81.1	0	0	0
AU 25	0	0	0	0	0	0	0	0	0	0	0	0	9.7	2.8	5.9	84.1	9.1	5.1
AU 26	0	0	0	0	0	0	0	0	0	0	0	0	3.8	5.3	9.4	12.7	81.1	6.1
AU 27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17.1	11	81

7.4.3 AUs Combinations Recognition for Image Sequences Containing AUs Combinations

In many cases AUs occur in combinations, an AU recognition system must have the ability to recognize both if they occur either single or in combinations of AUs. In previous work, AU recognition systems only were trained and tested on single AUs [25] [26] [27].

Because input image sequences contain single AUs and AUs combinations, the outputs have several possible outcomes. We can not only use Recognition Rate to show the results. So in this experiment three more standard metrics were used to evaluate the experiment. *Correct* denotes

that all AUs contained in input image sequences are recognized completely. *Partially correct* denotes that not all AUs contained in input image sequences are recognized completely but one or more than one AUs were recognized. In such case, the missed AUs or that AUs were added because of misrecognized (extra AUs) were also considered. Of course, if none of the AUs contained in input sequences were recognized, the result was *incorrect*.

In the third experiment, we use a different BN model as recognizer. It is not composed of some sub-models, which contain more than one ROI. Figure 6.3 is the structure of the BN model for recognizing AUs combinations. The model presents some AUs combinations contain those AUs with high occurrence.

Table 7.7: AUs Combinations Recognition for Image Sequences Containing AUs combinations.

Action Units	Recognized AUs								
	AU1+2	AU1+4	AU1+2+4	AU1+2+5	AU25+27	AU6+12	AU12+25	Practically Correct	
								Missing AUs	Extra AUS
AU1+2	89.7	21.5	26.9	12.8	0	0	0	0	12.2
AU1+4	21.3	88.9	15.9	12.1	0	0	0	0	13.4
AU1+2+4	21.4	22.4	89.4	31.5	0	0	0	11.3	0
AU1+2+5	23.2	22.5	27.9	89.9	0	0	0	17.9	0
AU25+27	0	0	0	0	88.6	10.2	36.9	11.3	27.9
AU6+12	0	0	0	0	12.5	86.2	26.7	10.9	21.8
AU12+25	0	0	0	0	21.9	11.8	85.1	20.8	31.8

7.5 Experimental Evaluations of AUs Recognition

We compare the testing results of our system in this thesis with some other AU recognition systems to evaluate the performance of our system. We compared our system with that of AFA system, Cohn et al., Lien et al., Bartlett et al., and Donato et al. The comparisons are summarized in Table 7.8. We choose some characteristics as standards to compare those systems which are recognition rates, feature detection methods, single AU recognized, AU combinations recognized and databases chosen.

Table 7.8: Comparison with Other AU recognition Systems.

System	Methods	Recognition rates	Treatment of AU combinations	AUs to be recognized	Databases
AU recognition system	Feature-detection	88.7%	YES	AU 1,2,4 AU5,6,7	Cohn-Kanade
	Optic-flow	87.5%	YES	AU9,10,12,15 AU17,20 AU23,24,25,26,27	
		89.1%	YES	AU1+2+4,AU1+2 AU1+4	
		84.2%		AU6+12,AU15+12 , AU25+27,AU12+2 5	
Current AFA system	Feature-based	88.5% (old faces)	No	AU 1, 2, 4, AU 5, 6, 7.	Ekman-Hager
		89.4% (novel faces)			
		95.4%	Yes/Yes	AU 9, 10, 12, 15, AU 17, 20 25, 26, AU 27, 23+24.	Cohn-Kanade
		95.6%	Yes/Yes		
Bartlett et al	Feature-based	85.3% (old faces)	No	AU 1, 2, 4, AU 5, 6, 7.	Ekman-Hager
		57% (novel faces)			
		Optic-flow			
	Hybrid	90.0%			
Donato et al	ICA or Gabor	96.9%	No	AU 1, 2, 4, AU 5, 6, 7.	Ekman-Hager

	Wavelet				
	Others	70.3%- 85.6%			
	ICA or Gabor	95.5%	Yes/No	AU 17, 18, 9+25, AU 10+25,16+25.	
	Others	70.3%- 85.6%			
Cohn et al	Feature- Tracking	89%	Yes/No	AU 1+2, 1+4, 4, AU 5, 6, 7.	Cohn- Kanade(Subset)
		82.3%	Yes/No	AU 12, 6+12+25, AU 20+25, 15+17, AU 17+23+24, 9+17 AU 25, 26, 27.	
Lien et al	Dense-flow	91%	Yes/No	AU 1+2, 1+4, AU 4.	Cohn- Kanade(Subset)
	Contour- detection	87.3%			
	Feature- Tracking	89%	Yes/No	AU 1+2, 1+4, AU 5, 6, 7.	
	Dense-flow	92.3%	Yes/No	AU 12, 6+12+25, AU 20+25, 15+17, AU 17+23+24, 9+17.	
	Feature- Tracking	88%	Yes/No	AU 12, 6+12+25, AU 20+25, 15+17, AU 17+23+24, 9+17, AU 25, 26, 27.	
	Contour- detection	80.5%	Yes/No	AU 9+17,12+25.	

Chapter 8

Emotion Recognition by Expert System

Nowadays a lot of research focuses on the detection of emotions from several kinds of input, like speech, text and visual input. Systems that recognize emotions from camera recordings have already been developed. The researchers' efforts to enable computers to detect human emotions will end up in providing a more and more natural human – machine communication, which is of interest of my thesis.

In the last implementation part of the thesis we built a system that would detect emotions from a facial expression. We still consider six basic emotions: happiness, sadness, anger, fear, disgust and surprise. The first step is to design a theoretical model. After designing the model, we need to set rules for our expert system, and then implement those using CLIPS.

This chapter outlines the implementation details of the developed expert system, capable of distinguishing between several scenarios based on a set of active features. Before getting into the intricate details of the produced solution, the utilized programming environment will be examined. Finally, the system's structure and principles are regarded and the reader is provided with the exact programming code the product is composed of.

8.1 Expert System

This chapter's goal was to develop an expert system suitable for recognizing emotions according to AUs rules. The solution to be developed was required to distinguish between several pre-defined scenarios which are the expression of six basic emotions, based on a selected set of activated AUs. This paragraph describes the general structure on which the developed expert system operates. First, the internal representation of scenarios is regarded, followed by a discussion of the system's algorithms, referring to the underlying programming code.

In order to realize such behavior in terms of an expert system, the first step is to properly define the various scenarios and their properties in a manner that is of use to the application. Each scenario is internally represented using several properties: a uniquely identifying title and two sets of feature ids positively and negatively influencing a hypothesis of the scenario.

As an example, consider the scenario definition displayed in Table 8.1. Aside from the scenario's title and id, it can be observed that the AUs that are required to be activated before adding the scenario as a hypothesis are those indicated by ids 1 and 37, respectively corresponding to the features descriptions "Lip corners go back and upwards" and "mouth open a bit". Internally, this information is stored using template-based facts. In the case of the example above, the following fact is added to the knowledge base:

```

(scenario
  (title "Scenario 1 - Happiness")
  (id      1
   )
  (main   1 37
   )
  (supporting 2,3,4,5,6
   )
  (weights 1 7 2 1 3
   ))

```

Table 8.1: This table displays an example of a scenario of happiness definition.

Scenario definition	
Id	1
Title	Scenario 1 – HAPPINESS
Main feature ids	{ 1, 37 }
Supporting feature ids	{ 2,3,4,5,6 }
Supporting feature weights	{ 1, 7, 2, 1, 3 }

A scenario’s set of weights correspond to its supporting features and represent their relative importance. For example, in the case of the illustrated scenario above, supporting feature 3 is 7 times as influential as feature 2. Before continuing the description of the implementation details with the next process in the system, it is necessary to support AUs by rules of each scenario, coupled with their associated weights. For example, Table 8.2 shows the supporting AUs for facial expression of happiness and the weights of each AUs.

Table 8.2: The supporting AUs for facial expression of happiness and the weights of each AUs.

Scenario 1: Happiness	
Supporting Features	Weights
1. teeth is visible (baring teeth)	1
2. lower eyelid shows wrinkles below it	7
3. lower eyelid be raised but not tense	2
4. wrinkles around the outer corners of the eyes	1
5. cheeks are raised	3

In a further phase of the expert system’s cycle, the above weights will be utilized to determine the degree in which scenarios’ hypotheses are considered confirmed by their supporting features. Before discussing this procedure in a more detailed fashion however, the following will continue the sequential description of the application’s internal processes.

Having initialized the expert system, control is passed to the select-features rule, which is responsible for reading and processing the user’s keyboard input, indicating the features that are desired to be activated. During this process, only low-level features are allowed to be activated. Then thus activated AUs are added to the fact database, in addition to a tracking fact, which

serves to keep count of the number of scenarios processed in a later step of the implemented system.

After activating the appropriate AUS, the combine-AUs rule is executed to ensure activation of higher-level features whenever necessary. The expert system contains a pre-defined list of the feature ids of higher-level features and the associated lower-level features they are comprised of. The process of combining features iterates over this list and checks whether or not all of a higher-level feature's lower-level components are activated. If this is the case, the higher-level feature is added to the list of active features. This process continues until no additional features are enabled. This approach guarantees a correct handling of higher-level features having lower-level components that are themselves higher-level features.

Once the final set of features has been determined, the initial selection of hypotheses can be performed. This process is accomplished through means of the select-hypotheses rule, which sequentially regards all the available scenarios, making use of the tracking facts that were added to the database in an earlier stage. In addition, in order to facilitate the next rule to be processed, a fact is utilized to keep count of the number of hypotheses that are added in this phase of the expert system's execution.

During the actual consideration of each scenario in this fashion, it is checked with its definition and the set of currently activated features whether or not all of the scenario's main features are active in the present state. If this is the case, the scenario becomes the latest addition to the initial set of hypotheses. Internally, this is accomplished by adding a (`supporting ?id`) fact to the database, (`where ?id`) represents the id of the scenario that was being regarded.

When the initial set of hypotheses is finalized, these facts enable the expert system to iterate over the individual hypotheses when searching for supporting features. This task is completed by the `find-supporting-AUs` rule. During its execution, this rule determines the activated supporting features for each scenario that's being considered and adds the associated weights. Next, the outcome is divided by the total sum of all the supporting features' weights in order to obtain the scenario's normalized 'supported value', indicating the degree to which the scenario's confirmed through means of its supporting features. The resulting value for each hypothesis is stored using a `scenario-score` fact. The final step in the expert system's progression is the output of the scenario, which is facilitated by the `output-best` rule.

The order in which the rules are executed is largely managed by a state fact, which is a container for the current state, described in a single keyword. For rules of an iterative nature, such as the `find-supporting-features` rule, tracking facts are used. For example, the `find-supporting-features` rule is triggered using the supporting facts, among others, which are added for each scenario that is part of the initial set of hypotheses. Furthermore, the number of hypotheses already regarded is tracked and continuously compared to the total amount of hypotheses present. Once they are found to be equal, the task to be performed by the rule has been accomplished and the state fact is modified to transfer control to the next process.

8.2 Implementation

In this chapter we present the implementation details underlying the developed solutions. First, we will construct CLIPS-based expert system to distinguish between rules based on some activated features that have been defined in previous sections.

8.2.1 CLIPS-Based Expert System

This paragraph outlines the implementation aspects of the expert system as developed in CLIPS. In the remainder of this section we will provide the intricate details of the development model's internal workings. We will give a closer examination of the rules the expert system is founded on and expose the lower-level details of the application.

8.2.1.1 Implementation Details

In this paragraph we proceed to describe the implementation details through analyzing the expert system's procedure. The procedure can be performed by following separate steps which is regarded in a sequential fashion, highlighting their underlying details.

When the expert system is launched, the first step is initialize rule, based on the initial fact as provided by CLIPS upon resetting its environment. This initialization rule adds a set of trigger facts to the database, which is needed to properly loop over the scenarios in a later stage of the application cycle. Furthermore, each definition of scenario should be added to the system. Each such definition contains a scenario's title, main features, supporting features and their associated weights.

Having initialized the expert system, which is responsible for reading and processing the user's keyboard input, indicating the features that are desired to be activated, and using select-features rule to choose low-level features. It notes that during this process, we just can only active low-level features. Such domain theories typically use concepts which are not precisely defined, and deal with phenomena that are imperfectly understood. Then add the activated features to the fact database, in addition to a tracking fact, which serves to keep count of the number of scenarios processed in a later step of the implemented system.

After activating the appropriate features, we execute the combine-features rule to ensure activation of higher-level features whenever necessary. The expert system contains a pre-defined list of the feature of higher-level features and the associated lower-level features they are comprised of. The process of combining features iterates over this list and checks whether or not all of a higher-level feature's lower-level components are activated. If this is the case, the higher-level feature is added to the list of active features. We should loop the process until no additional features are enabled. This approach guarantees a correct handling of higher-level features having lower-level components that are themselves higher-level features.

Once the final set of features has been determined, the initial selection of hypotheses can be performed. This process is accomplished through the select-hypotheses rule, which sequentially regards all the available scenarios, making use of the tracking facts that were added to the database in an earlier stage. In addition, in order to facilitate the next rule to be processed, we use a fact to count the number of hypotheses that are added in this phase of the expert system's execution.

During the actual consideration of each scenario in this fashion, it is checked with its definition and the set of currently activated features whether or not all of the scenario's main features are active in the present state. If this is the case, the scenario becomes the latest addition to the initial set of hypotheses. Internally, this is accomplished by adding a (supporting? id) fact to the database, (where? id) represents the id of the scenario that was being regarded.

When we finalize the initial set of hypotheses then we use these facts to enable the expert system to iterate over the individual hypotheses during the process of searching for supporting features. This is responsible to complete the find-supporting-features rule. During its execution, this rule determines the activated supporting features for each scenario that's being considered and adds the associate weights. Next, in order to obtain the scenario's normalized 'supported value' we divide the total sum of all the supporting features' weights. We use "supported value" to indicate the degree to which the scenario's confirmed through means of its supporting features. The resulting value for each hypothesis is stored using a scenario-score fact.

Having determined the supported value for each hypothesis of the initial set of hypotheses, we use the select-maximum rule to establish the maximum- scoring scenario over the associated facts.

At the final step in the expert system's progression we should execute output-best rule to facilitate the output of scenario.

How to order the execution of the rules is largely managed by a state fact, which contains the current state and is described in a single keyword. For rules of an iterative nature, such as the find-supporting-features rule, tracking facts are used. For example, the find-supporting-features rule is acted by the supporting facts, among others, which are added for each scenario that is part of the initial set of hypotheses. Furthermore, we track and continuously compare the number of hypotheses already regarded to the total amount of hypotheses present. As long as we find them to be equal, we can say the task to be performed by the rule has been accomplished and the state fact is modified to transfer control to the next process.

From above discussion we have presented the implementation of the developed CLIPS-based expert system's internal procedures. In addition, the way in which they are structured to form a fully functioning application also has been concluded in this section.

8.2.1.2 Testing

This paragraph will describe the testing process as performed for the application that was developed in CLIPS. Testing was as straightforward as with the previously implemented CLIPS-based expert system. The application was designed in such a fashion as to output all its processing steps, allowing clear-cut evaluation and validation of its decisions. An example of the output associated with these steps is provided in Listing 8.1 and further, as displayed below.

In practice, validating the application entails running the expert system, while a test user is providing the required input, and manually checking whether or not the console output matches the desired output. Because of the algorithm's simplicity, a human is very well capable of performing this final verification step. In a similar fashion, any unexpected and undesired results were easily mapped onto flaws in the underlying programming code, which were subsequently fixed.

Having discussed the testing and validation process from a general point of view, it is now time to examine a specific practical example of a single run of the expert system, as displayed in Listing 8.1 through Listing 8.3 below.

```
CLIPS> (reset)
CLIPS> (run)

Available features:
-----
                        Happiness
-----

Main features:
1. Lip corners go back and upwards
37 mouth open a bit

Supporting features:
2. teeth is visible (baring teeth)
3. lower eyelid shows wrinkles below it
4. lower eyelid be raised but not tense
5. wrinkles around the outer corners of the eyes
6. cheeks are raised

Specify a list of ids of the features to activate.
Entries should be separated by spaces.
Example: 1 5 9 3 12
1 20 9 5 7 16 25 28 19 17 29
```

Listing 8.1: The expert system initially requires the user to set the features that are to be activated.

First, the application outputs all the available features to the user. This allows the user to read through them whilst selecting the features that are of interest. After the user makes a selection the system displays the activated features and allows the user to start the analysis at the press of any key. When the user finally chooses to start the analysis the application starts searching through the activated features and extracts the ones that serve as main features to certain scenarios. The activated main features and their corresponding scenarios are made visible as are all the other scenarios and their main features.


```

Activated feature 1 (Lip corners go back and upwards)
Activated feature 37 (mouth open a bit)
Activated feature 2 (teeth is visible (baring teeth))
Activated feature 5 (wrinkles around the outer corners of the eyes)
Activated feature 7 (Normal voice [multiple])
Activated feature 16 (brows are raised)
Activated feature 25 (open eyes wide )
Activated feature 28 (lower lip is pushed up to upper lip )

Press any key to start the analysis.

CHECKING FOR HIGH-LEVEL FEATURE ACTIVATIONS

High-level feature 'Happiness' (1) was activated.
All of its lower-level features were previously activated:
 3 lower eyelid shows wrinkles below it
 4. lower eyelid be raised but not tense
 6. cheeks are raised

```

Listing 8.2: After activating higher-level features, the system selects the initial set of hypotheses.

Next, the system finds what supporting features of the activated scenarios are also activated and assigned a score for every active emotion and displays the emotion with the highest score. The entire process of finding supporting features and the assigning of scores to emotion is fully transparent to the user through means of continuous output.

```

FINDING SUPPORTING FEATURES FOR HYPOTHESES

Regarding hypothesis 'scenario 1 - Happiness' with the following supporting features.
 2 teeth is visible (baring teeth) (weight 1 ) (inactive)
 3 lower eyelid shows wrinkles below it (weight 7) (active)
 4 lower eyelid be raised but not tense (weight 2 ) (inactive)
 5 wrinkles around the outer corners of the eyes (weight 1 ) (inactive)
 6 cheeks are raised (weight 3 ) (active)
Processing active weights...
Raw 7, normalized: 0.625
Raw 3, normalized: 0.1875
The scenario has been awarded a normalized score of 0.8125
(weights were normalized through means of a division by the total of
all supporting features' weights, which was 16)

```

Listing 8.3: This listing illustrates the search for features supporting the initial set of hypotheses and the determination of the dominant hypothesis.

8.3 Results Evaluation

Altogether we have rendered 7 different expressions in poser, edited the files in FireWorks and passed the data in Matlab, providing us 6 different **.clp** files to use in CLIPS (only six, since the neutral expression is included as a reference for all the others). We have Anger, Fear, Happiness, Sadness, Surprise, Disgust... We will shortly discuss three of the tests here. All the others can be reproduced by running CLIPS.

Anger (anger.clp)

The **anger.clp** file has manually been created using the output from the editing the rendering in FireWorks, and passing the image to Matlab. The image used to retrieve the data for this specific emotion. Loading the **emotion.clp** along with the **anger.clp** files produces the following results. We see that the anger emotion is recognized successfully.

Emotions		Found AU' s
Anger	38%	AU17
Disgust	0%	AU26
Fear	0%	AU23
Happiness	0%	AU4
Sadness	14%	
Surprise	20%	

Found Emotion: Anger is correct

Fear (fear.clp)

The **fear.clp** file has manually been created using the output from the editing, and passing the image to Matlab. Loading the **emotion.clp** along with the **anger.clp** files produces the following results. We see that the fear emotion is recognized as Surprise which is not correct.

Emotions		Found AU' s
Anger	8%	AU1
Disgust	43%	AU17
Fear	0%	AU26
Happiness	0%	AU9
Sadness	29%	AU2
Surprise	60%	

Found Emotion: Surprise is incorrect

8.4 Summary

Our implementation currently performs reasonable on detecting the emotions from the images. The results heavily depend on the displayed emotions. In case of pure emotions they will be recognized correctly. But in most cases facial expressions show blended emotions. Even human labelers have problems in labeling such emotions. Flaws can arise in many parts of the process. Starting with, inserting the data in CLIPS. Also the rules depend on threshold values which values would have to be investigated further to be able to produce more accurate results. Finally

flaws exist in converting the activated AUs to an actual emotion. The function for selecting the final emotion is a very basic one which does not take any importance into account.

The most important conclusion, however, is that it is indeed possible to recognize an emotion from different face images using the procedures in this thesis. A lot of tweaking is required to achieve better results though. The question remains whether or not an expert system is a good approach to retrieve emotions from a face. We can indeed conclude that an expert system is a good way in doing so, but CLIPS might be not the best tool to achieve good results. Retrieving the data can be improved to an acceptable level. Also the threshold values could be studied to produce better results.

But the final decision on which emotion is shown can cause serious problems. For good results in this part of the process, one would prefer to use probabilities. The right probabilities would greatly improve the performance of our implementation. But using CLIPS it is difficult to get the proper probabilities for this task. That is why we choose BN to recognize AUs, but in our case because of the labeling problem it is not necessary to recognize emotions by BN for Cohn-Kanade dataset. In future work, we will choose another dataset which has the correct labels to recognize emotions through BN.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

In this section, eventually we can provide a final discussion of the problems to be solved. Here, we reflect on the development process in general and the obtained results in particular. We also should regard the development of the CLIPS-based expert system. Finally, the Bayesian network part of the assignment is reflected upon. We use three layer networks with one hidden layer to recognize AUs by a Bayesian Networks. The inputs are the two parameters in each region and the outputs are 18 signal AUs and some AUs in combination.

We plan to describe the whole process of our work in the following before getting into a reflection of the developed applications. Since we have known a long history of co-developing solutions to projects and practical assignments, a flexible and efficient working environment was both expected and actualized. We really faced a lot of difficulties which was largely due to the fact that we did not have much knowledge about optical flow algorithm. Bayesian Network is not strange for us, but it is the first time for to design a good module to solve a problem. During the process we really learned a lot. After these initial struggles, we rapidly designed the BN module and chose the parameters from detected features. But through considering the complexity of network and the changing of parameters, the module and parameters have been improved many times. We really spent slightly longer to construct the Bayesian network as the arranging and combining of the different feature nodes into separate intermediate nodes and finally into the stress-o-meter node proved to be a challenging task.

In the next paragraphs first we review our initial research goals which have been presented in chapter 4 then according to the goals to show to what extent they were satisfied. Finally, we present five opportunities for future work.

9.1.1 Goals and Objectives

Before giving the conclusion of the thesis, an overview of the research questions and objectives are necessary to be presented in order to follow a one-by-one discussion of the satisfaction of these goals and objectives:

- ✓ **Feature Extraction**
 1. Optical flow is detected by Lucas-Kanade method.
 2. How to analysis those optical flows extracted.
 3. Find out the parameters can be used as the input in classification.

- ✓ **Modeling Facial Expressions**
 1. How many levels in this module and what is the relationship between them.
 2. Training data in a model or in each sub-model.
 3. The out-put of the model is six basic emotions directly or not.
 4. If in this model only recognize AUs then how to recognize six basic emotions.

- ✓ **Facial Expression Recognition**
 1. How to active multisensory information fusion in a framework of dynamic.
 2. How to assign the dataset to training and testing.
 3. How to recognize six basic emotions by AUs.

Based on the above questions, we formulated our assignment. The original assignment was designed three main parts: Facial Motion Measurement, Facial Expression Representation and Facial Expression Recognition. In their turn, these three main research fields consisted of our goals and objectives of the research presented in this thesis.

9.1.2 Facial Motion Measurement

The measurement of facial motion is through tracking of facial features by optical flow algorithm. And in this thesis one of the main contribution is recognize single AU and AU combinations so how to detect facial landmarks was a research topic. Meanwhile both how to decide the region of our interested and how to analysis the optical flows extracted in those regions are main research goals.

1. Automatic Detection of Facial Landmarks from AU-Coded expressive facial images

We first implemented a feature-based method for expression-invariant localization of facial landmarks from static images. Based on the contour map method and contour orientation matching algorithm we applied local contrast thresholding calculated in every filter neighborhood instead of using an average contrast of the whole image to define thresholds fro contrast filtering. Then according to the orientation of eye, eyebrows and mouth and the knowledge of a face geometry model to compute the candidates and distance to localize the nose, lips and mouth corners.

2. Defining the regions of our interested (ROI)

We used a popular algorithm to analysis the motion of vector flows which is Entropy Maximum algorithm because entropy is a good way of representing the impurity or unpredictability of a set of data since it is dependent on the context in which the measurement is taken. According to the ROI to analysis the optical flows and computes the parameters as the input in classification is a good approach because AUs are related to different parts of face.

3. Optical Flow Analysis by compass diagram in ROI

We implemented a Matlab program to convert those optical flows extracted to a compass function vector diagram to show their summation of the directional vectors. That is to say we created representation of the whole face from the quiver diagram. In the compass diagram summed all the vectors from the quiver diagram and created a determination of the total directional vectors from the centre of the origin. It shows the major directions and velocities of facial movement. First we only map the optical flows detected from the whole face to compass diagram but we found differences among six basic emotions is not so significant. Then we map those optical flows detected from ROI to compass diagram and the result is we expected. For different emotions in the same ROI the compass vectors shown are significant different. That is to say the parameters we used to training has the obvious different for different emotions.

9.1.3 Facial Expression Representation

We designed a BN model of facial expression which consists of three primary layers, namely, emotion expression classification layer contains 6 basic emotions, optical flow data layer contains ROI, and facial AU layer.

4. Facial Expression Model

The classification layer consists of a class (hypothesis) variable E including six states $e_1, e_2, e_3, e_4, e_5, e_6$, which represent six basic emotion (happiness, sadness, disgust, surprise, anger, and fear) respectively, and a set of attribute variables denoted as HAP, ANG, SAD, DIS, SUR, and FEA corresponding to the six facial expressions. The goal of this level of abstraction is to find the probability of class state e_i , which represents the chance of class state e_i given facial observations. When this probability is maximal, it has the largest chance that the observed facial expression belongs to the state of class variable e_i .

The optical flow level of layer in the model is the data layer containing optical flow information variables, which are the lowest level of layer in our model. We defined nine interested regions in which the optical flows have been detected by Lucas and Kanade algorithm. The nine ROI are Brows, Lips, Lip Corners, Eyelids, Cheeks, Chin, Mouth, Nasolabial Furrow, and Wrinkles, which contains the optical flow information variables which are $\overline{V}_x, \overline{V}_y$ and average angle for each region.

The level of the layer is the AU layer is analogous to linguistic description of the relation between AUs and facial expressions. And for every emotion expression in the related ROI the AUs that are related the six facial expressions have been given.

5. Learning Parameters

After given the BN structure we need to learn the parameters in order to infer each AU. First a dataset labeled according to the parameters estimated from our feature extraction and automatically facial region localization system has been prepared. Then EM algorithm was applied in GeNIe to train those dataset. The goal of learning parameters is to maximize the posterior distribution so the initial probability of each parameter should be given after applying EM algorithm the CPT is obtained. This learning process considers both prior probability and likelihood so that the posterior probability is maximized since the training dataset is complete.

6. AU Recognition through Inference

We conducted three experiments to evaluate the inference. The first is about recognizing single AUs when the image sequences contain only a single AU. 18 important AUs are recognized with the highest recognition rate. In the second experiment, we also used the same BN model as in the first experiment, and the inputs and outputs are also the same. The difference is that images sequences were trained and tested which not only contain the single AUs but also contain AUs combinations. The average of recognition rate is higher than in the first experiment because for single AUs there are not enough data even for some AUs only have fewer than 6 images. In the third experiment, AU combinations are recognized. Because input image sequences contain AUs and AU combinations, the outputs have several possible outcomes. In such case, the missed AUs or that AUs were added because of misrecognized were also considered. The recognition rate is higher than 85%.

7. Emotion Recognition by ES

In the last level we did not use the same approach to extract emotions from related AUs rather than to use a simple expert system by programming in CLIPs to extract the emotions. Because there is a problem of how to use the Cohn Kanda database. The video only provides the list of activated and deactivated AU's but doesn't provide labels for emotions. In fact some videos do not show any emotion at all but is just to show the impact of one or more AU's. In Philips Research they were using a subset and attach some labels to the database, but we have the doubts about the validity of the labels. But the labels with AUs in the database are correct and done by Prof Cohn himself. So we have to say the database is only suitable for recognizing AUs instead of emotions.

9.2 Future Work

Research on recognizing emotion modeling with BNs is currently a very active field. And in fact, in recent research more and more researchers focused on multimodal emotion recognition which is also our research goal in MMI group in TUD. In general, the research goal for emotion recognition is to fuse different features detected from different channels, in which the most important step is to combine feature and decision levels. Dynamic Bayesian Networks are a good

structure for fusing features and fusing channel, in which we can give different weightings for every layer's features. As such, many opportunities for future research exist.

Based on the research reported in this thesis, more opportunities for future research in this field can be summarized in four parts: application in market, database, and feature detection and fusion approach.

First of all, the final system can be applied to the consumer market; for this research goal, we can categorize emotions into passive/active and negative/positive, because we just need to know whether the consumer likes or dislikes a product - a sentiment detection exercise. If at first the emotion can be recognized as passive then I will not continue to further recognize what the emotion is. If at first the emotion can be recognized as active, that is to say the person has a feeling about this product, and then we continue to recognize whether the emotion is negative or positive.

A second opportunity for future work arises in "naturalness" database. The easiest way to collect emotional speech is from actors. However, acted speech is often "read", not spoken, and read speech is well known to have distinctive characteristics. Actors may also show the obvious facial and body language that would not happen in daily life, especially in public places. So the ideal data will be from truly natural behavior.

A third opportunity for future work is to analyze features from more channels instead of only facial expression or speech signal. The vision fusion can be analyzed from facial expressions, head movement, gaze direction and upper-body, and speech signals fusion can be analyzed from tone, marked voice and words. In addition, the ideal system should employ tightly coupled audio and visual modalities; moreover, the multimodal signals should be considered mutually dependent rather than be combined only at the end as is the case in decision-level fusion.

The last but not least challenge is a common open issue for recent research on multimodal emotion recognition, regarding when and how to combine the uni-modal to a multimodal. The same problem also includes in classifying features such as when and how to combine the features from facial, speech and gesture analysis. The main issue focus on what is the optimal level of combining different streams, how to get the optimal function for the combination and remain the reliability of each stream in which each feature detected depends on each other.

Finally, the last opportunity for future work is to train and test AUs for six basic emotion classes separately. If we have a database in which all images have been labeled emotions then we can use the same BN model as in the first and second experiment, and the inputs and outputs are also the same. And the images sequences were trained and tested are not only contain the single AUs but also contain AUs combinations. But the images sequences were assigned according to six basic facial expressions. And the training data is from each class.

Bibliography

- [1] Yongmian Zhang and Qiang Ji, "Active and Dynamic Information Fusion for Facial Expression Understanding from Image Sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, No. 5, Pages: 699-714, 2005.
- [2] Jenn-Jier James Lien, "Automatic Recognition of Facial Expressions Using Hidden Markov Models and Estimation of Expression Intensity," *PHD Thesis, The Robotics Institute Carnegie Mellon University Pittsburgh, Pennsylvania*, 1998,
- [3] D. Datcu and L.J.M. Rothkrantz, "Automatic recognition of facial expressions using Bayesian Belief Networks," *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 7, Pages: 2209-2214, 2004.
- [4] Jeffrey F. Cohn, Adena J. Zlochower, James Lien and Takeo Kanade, "Automated face analysis by feature point tracking has high concurrent validity with manual FACS coding," *Society for Psycho physiological Research*, Pages 35–43, 1999.
- [5] Yingli Tian, Takeo Kanade and Jeffrey F.Cohn, "Recognizing Action Units for Facial Expression Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 23, No.2, 2001.
- [6] Stuart Russell, John Binder, Daphne Koller and Keij Kanazawa, "Local learning in probabilistic networks with hidden variables," *Computer Science Division University of California Berkeley, CA, USA*.
- [7] Jiří Vomlel, "Noisy-Or Classifier," *Institute of Information Theory and Automation Academy of Sciences of the Czech Republic Prague, Czech Republic*.
- [8] Irfan A. Essa, "Coding Analysis Interpretation, and Recognition of Facial Expressions. College of Computing," *GVU Center, Georgia Institute of Technology, Atlanta, GA, USA*, 1997.
- [9] Yan Tong, Wenhui Liao and Qiang Ji, "Facial Action Unit Recognition by Exploiting Their Dynamic and Semantic Relationships," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, No. 10, Pages 1683-1699, 2007.
- [10] Jeffrey F. Cohn, James J. Lien and Adena J. Zlochower, "Feature-Point Tracking by Optical Flow Discriminates Subtle Differences in Facial Expression," *Department of Electrical Engineering University of Pittsburgh*.
- [11] Irfan A. Essa and Alex P. Pentland, "Facial Expression Recognition using a Dynamic Model and Motion Energy," *International Conference on Computer Vision '95*,

Cambridge, MA, 1995.

- [12] Zhu Yongli, Huo Limin, and Lu Jinling, “Bayesian Networks-Based Approach for Power Systems Fault Diagnosis,” *IEEE Transactions on Power Delivery*, Vol. 21, No. 2, 2006.
- [13] Agnieszka Oni´sko, Marek J. Druzdzal and Hanna Wasyluk, “Learning Bayesian Network Parameters from Small Data Sets: Application of Noisy-OR Gates,” *Working Notes of the Workshop on ‘Bayesian and Causal Networks: From Inference to Data Mining,’ 12th European Conference on Artificial Intelligence (ECAI-2000)*, Berlin, Germany, 2000.
- [14] Fabio G. Cozman, “Axiomatizing Noisy-OR”.
- [15] Tian Fengzhan, Xu Ke, Zhang Hongwei and Lu Yuchang, “A Novel Learning Method for Special Bayesian Networks,” *Proceedings of the 4th World Congress on Intelligent Control and Automation*, Shanghai, P.R.China, 2002.
- [16] Guifen Chen and Helong Yu, “Bayesian Network and ITS Application in Maize Diseases Diagnosis,” *IFIP International Federation for Information Processing*, Vol. 259, Pages 917–924, 2008.
- [17] Benjamin Hernandez, Gustavo Olague, Riad Hammoud and Leonardo Trujillo Eva Romero, “Visual Learning of Texture Descriptors for Facial Expression Recognition in Thermal Imagery,” *Computer Vision and Image Understanding*, Vol. 106, Pages 258-269, 2007.
- [18] Leonardo Trujillo, Gustavo Olague, Riad Hammoud and Benjamin Hernandez, “Automatic Feature Localization in Thermal Images for Facial Expression Recognition,” *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, 2005.
- [19] Satyabrata Pradhan, Rajveer Singh, Komal Kachru, Srinivas Narasimhamurthy, “A Bayesian Network based Approach for Root-cause-analysis in Manufacturing Process,” In *International Conference on Computational Intelligence and Security*, 2007.
- [20] J.L.Barron and N.A.Thacker, “Tutorial: Computing 2D and 3D Optical Flow,” www.tina-vision.net, 2005.
- [21] Ilse Ravyse, Dongmei Jiang, Xiaoyue Jiang, Guoyun Lv, Yunshu Hou, Hichem Sahli, and Rongchun Zhao, “DBN Based Models for Audio-Visual Speech Analysis and Recognition,” *PCM 2006*, LNCS 4261, pages19–30, 2006.
- [22] Jeffrey F. Cohn, Adena J. Zlochower, James Lien, and Takeo Kanade, “Automated Face Analysis by Feature Point Tracking Has High Concurrent Validity with Manual FACS Coding,” *Department of Psychology, University of Pittsburgh and Robotics Institute, Carnegie Mellon University*.

- [23] Robert Niese, Ayoub Al-Hamadi, Axel Panning, and Bernd Michaelis, “Real-Time Capable Method for Facial Expression Recognition in Color and Stereo Vision,” *ICCSA 2007, LNCS 4705, Part I*, pages 397–408, 2007.
- [24] Prem Kalra, Angelo Mangili, Nadia Magnenat-Thalmann, Daniel Thalmann, “SMILE: A Multilayered Facial Animation System,” *Computer Graphics Lab MIRALab, CUI Swiss Federal Institute of Technology University of Geneva, Switzerland*, 1991.
- [25] Susanne G. Böttcher Claus Dethlefsen, “Learning Bayesian Networks with R,” *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*, Vienna, Austria, March 2003.
- [26] Carmen J. Duthoit T. Szynda, S.K.L. Lal, B.T. Jap and J.I. Agbinya, “Optical Flow Image Analysis of Facial Expressions of Human Emotion – Forensic Applications,” *e-Forensics 2008*, Adelaide, Australia, January 2008.
- [27] Yaser Yacoob and Larry S. Davis, “Recognizing Human Facial Expressions from Long Image Sequences Using Optical Flow,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 6, June 1996.
- [28] Yan Tong, Wenhui Liao and Qiang Ji, “Facial Action Unit Recognition by Exploiting Their Dynamic and Semantic Relationships,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, No. 10, October. 2007.
- [29] Carole R. BEAL and Lei QU, “Relating Machine Estimates of Students’ Learning Goals to Learning Outcomes: A DBN Approach,” *R. Luckin, K. R. Koedinger, & J. Greer (Eds.), Artificial intelligence in education: Building technology rich learning contexts that work*, Pages: 111-118, Amsterdam: IOS Press.
- [30] Thomas Hamelryck, The Mocapy Toolkit, “Inference and Parallelized Learning in Dynamic Bayesian Networks,” *Bioinformatics Center University of Copenhagen Universitetsparken 15, bygning 10 2100 Copenhagen Denmark*, Version 0.55, November 2004.
- [31] M. Jaeger, Decision Support Systems and Machine Learning, “Lecture 4: Bayesian Networks – Modeling techniques,” Aalborg University.
- [32] Nicu Sebe, Ira Cohen, Theo Gevers, and Thomas S. Huang, “Multimodal Approaches for Emotion Recognition: A Survey, Faculty of Science, University of Amsterdam,” The Netherlands, HP Labs, USA, Beckman Institute, University of Illinois at Urbana-Champaign, USA.
- [33] Maja Pantic and Leon J. M. Rothkrantz, “Toward an Affect-Sensitive Multimodal Human–Computer Interaction,” *IEEE Invited Paper*, Vol. 91. Pages 1370-1390, 2003.

- [34] Ira Cohen, Fabio G. Gozman, Marcelo C. Cirelo, “Learning Bayesian network classifiers for facial expression recognition both labeled and unlabeled data,” *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, Vol. 1, Pages: I-595- I-601, June 2003.
- [35] Rana El Kaliouby and Peter Robinson, “Real-Time Inference of Complex Mental States from Facial Expressions and Head Gestures,” *Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04)*, Vol. 1, Pages:154, 2004.
- [36] Ashish Kapoor and Rosalind W. Picard, “A Real-Time Head Nod and Shake Detector, Affective Computing,” *ACM International Conference Proceeding Series*, Vol. 15, Pages: 1-5, 2001.
- [37] Jari Kätsyri, “Human Recognition of Basic Emotions from Posed and Animated Dynamic Facial Expressions,” *Dissertation for the degree of Doctor of Philosophy to be presented for public examination and debate in Auditorium K at Helsinki University of Technology* December 2006.
- [38] Carmen J.Duthoit, T.Sztynda, S.K.L., Lai, B.T. Jap, J.I.Agbinya, “Optical Flow Image Analysis of Facial Expressions of Human Emotion—Forensic Applications,” *Proceedings of the 1st international conference on Forensic applications and techniques in telecommunications, information, and multimedia and workshop*, No. 5, 2008.
- [39] Benjamin Hernandez, Gustavo Olague, Riad Hammoud, Leonardo Trujillo, Eva Romero, “Visual Learning of Texture Descriptors for Facial Expression Recognition in Thermal Imagery,” *Computer Vision and Image Understanding*, Vol. 106 , Issue 2-3, Pages 258-269, May. 2007.
- [40] Yulia Gizatdinova and Veikko Surakka, “Automatic Detection of Facial Landmarks from AU-coded Expressive Facial Images,” *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on*, Pages: 419-424, September, 2007.
- [41] Ramesh Ram, Madhu Chetty, “Learning Structure of a Gene Regulatory Network,” *6th IEEE/ACIS International Conference on Computer and Information Science*, Pages: 525-531, July, 2007.
- [42] Jie Cheng, David Bell, Weiru Liu, “Learning Bayesian Networks from Data: An Efficient Approach Based on Information Theory,” <http://www.cs.ualberta.ca/~jcheng/Doc/report98.ps>. 1997.
- [43] Akt Kaban, “Lecture Slides: Introduction to Bayesian Learning,” *School of Computer Science, University of Birmingham*.
- [44] ZuWhan Kim, Ramakant Nevatia, “Learning Bayesian Networks for Diverse and Varying Numbers of Evidence Sets,” *Proceedings of the Seventeenth International Conference on*

Machine Learning, Pages: 479 - 486, 2000.

- [45] G. Valentini, R. Tagliaferri, F. Masulli, “Machine Learning with Bayesian Networks for Gene Function Prediction,” *Computational intelligence and machine learning in bioinformatics Artificial Intelligence in Medicine*, Vol 45, Issue 2, Pages 91-96.
- [46] Nir Friedman, Moises Goldszmidt, “Learning Bayesian Networks with Local Structure,” Stanford University Dept. of Computer Science, SRI International, 333 Ravenswood Way, EK329.
- [47] Carmen J. Duthoit, T.Sztynda, S.K.L.Lai, B.T.Jap, J.I. Agbinya, “Optical Flow Image Analysis of Facial Expressions of Human Emotion—Forensic Applications,” *Proceedings of the 1st international conference on Forensic applications and techniques in telecommunications, information, and multimedia and workshop*, No.5, 2008
- [48] Feigenbaum, E.A., “How the ‘what’ becomes the ‘how’,” *Commun. ACM* , Vol. 39(5), Pages. 97-104, May, 1996.
- [49] Friedman-Hill, E.J., Jess, “the Java Expert System Shell,” http://herzberg.ca.sandia.gov/jess/docsMarch_11,2000, Version 5.0. Sandia National Laboratories, Livermore, CA, 2000.
- [50] Giarratano, J.C., “CLIPS user’s guide: CLIPS,” Versio 6.0. NASA, Lyndon B. Johnson Space, TX 1993.
- [51] Giarratano, J. and Riley, G., “Expert Systems Principles and Programming,” Version 3rd PSW Publishing, Boston, MA, 1998.
- [52] Hartley, S., Gerhardt-Powals, J., Jones, D., McCormack, C., Medley, M.D., Price, B. Reek, M., and Summers, M.K. “Enhancing teaching using the Internet: Report of the working group on the World Wide Web as an interactive teaching resource,” *In Proceedings of the Conference on Integrating Technology into Computer Science Education*. Barcelona, Spain, June, 1996.
- [53] McFarland, T.D., and Parker, R, “Expert Systems in Education and Training,” Educational Technology Publications, Englewood Cliffs, NJ, 1990.
- [54] I.A. Essa and A.P. Pentland. “Coding, analysis, interpretation, and recognition of facial expressions,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7):757– 763, 1997.
- [55] B. Fasel and J. Luetttin. “Automatic facial expression analysis: A survey,” *Pattern Recognition Society*, Vol. 36, Pages: 259–275, 2003.
- [56] Raouzaïou, A., Tsapatsoulis, N., Karpouzis, K., Kollias, “Parameterized facial expression synthesis based on MPEG-4, EURASIP,” *Journal on Applied Signal Processing*, Vol. 2002, No 10, 2002, Pages. 1021-1038.

- [57] Sebe, N., Cohen, I., Huang, T.S. “Multimodal Emotion Recognition,” *Handbook of Pattern Recognition and Computer Vision*, World Scientific, ISBN 981-256-105-6, January 2005.
- [58] J.T. Cacioppo and L.G. Tassinary. “Inferring psychological significance from physiological signals,” *American Psychologist*, Vol. 45, Pages:16–28, 1990.
- [59] A. Garg, M. Naphade, and T.S. Huang. “Modeling video using input/output markov models with application to multi-modal event detection,” In B. Furht, O. Marques, and B. Furht, editors, *Handbook of Video Databases: Design and Applications*, 2003.
- [60] A. Garg, V. Pavlovic, and J. Rehg. “Boosted learning in dynamic Bayesian networks for multimodal speaker detection,” *Proceedings of the IEEE*, Vol. 91, Pages: 1355–1369, 2003.
- [61] D. Goleman. *Emotional Intelligence*. Bantam Books, 1995.
- [62] Pantic, M., Rothkrantz, L.J.M. “Automatic analysis of facial expressions: The state of the art,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 22, Pages:1424–1445 2000.
- [63] C.E. Izard. Innate and universal facial expressions, “Evidence from developmental and cross-cultural research. *Psychological Bulletin*, Vol. 115, Pages: 288–299, 1994.
- [64] C.C. Chiu, Y.L. Chang, and Y.J. Lai. “The analysis and recognition of human vocal emotions,” In *Proc. International Computer Symposium*, Pages: 83–88, 1994.
- [65] Camurri, A., Coletta, P., Massari, A., Mazzarino, B., Peri, M., Ricchetti, M., Ricci, A. and Volpe, G. “Toward real-time multimodal processing: EyesWeb 4.0,” In *Proc. AISB 2004 Convention: Motion, Emotion and Cognition*, Leeds, UK, March 2004.
- [66] Witten, I.H., Frank, E, “Data Mining: Practical machine learning tools and techniques,” *2nd Edition*, Morgan Kaufmann, San Francisco 2005.
- [67] L.S. Chen. “Joint processing of audio-visual information for the recognition of emotional expressions in human-computer interaction,” *PhD thesis, University of Illinois at Urbana-Champaign, Dept. of Electrical Engineering*, 2000.

Appendices A: Six Basic Emotion States Described by Action Units

The six basic emotion states are described by AUs. In addition to the “sufficient features”, “necessary features” which enable readers to gain a better understanding of the emotion states are necessary as well. Before proceeding to the analysis and extraction of the scenarios’ features, it was mandatory to decompose the scenarios in a storyboard-like fashion. The following present the results of this process, describing the six basic emotions in a point-wise user. Deeply remembering the emotion states which is important to causal the relationship between facial expressions and AUs.

Table A.1: Facial Expression Display: Happiness.

Facial Expression Display 1: Happiness
Beginning: Raising mouth corners or mouth opening with its expansion
1. sufficient: corners of lips are drawn back and up
2.necessary :mouth may be parted with teeth exposed (but for Chinese may be not)
3. sufficient :lower eyelid shows wrinkles below it Necessary :lower eyelid be raised but not tense
4. sufficient: wrinkles around the outer corners of the eyes
5. sufficient: cheeks are raised
Ending: lowering mouth corners or mouth closing with its contraction

Table A.2: Facial Expression Display: Sadness.

Facial Expression Display 2: Sadness
Beginning: lowering mouth corners, raising mid mouth and raising inner parts of brows
1.sufficient: inner eyebrows are drawn up
2.necessary: upper lid inner is raised
3.sufficient: corners of the lips are drawn downwards (maybe just a little bit)
Ending: lowering mouth corners, lowering mid mouth and lowering inner parts of brows

Table A.3: Facial Expression Display: Surprise.

Facial Expression Display 3: Surprise
Beginning: raising eyebrows and lowering of lower lip (jaw)
1. sufficient : skin below brow stretched but not wrinkled
2. sufficient : horizontal wrinkles across forehead
3. necessary: eyelids opened
4. sufficient: jaw drops open or stretching of the mouth
5. necessary: brows raised
Ending: lowering brows and raising of lower lip (jaw)

Table A.4: Facial Expression Display: Anger.

Facial Expression Display 4: Anger
Beginning: inward lowering brows and mouth compaction
1.sufficient : brows lowered and drawn together
2.necessary: lower lid is tensed , and be raised (maybe just a little bit)
3.sufficient: lines appear between brows means wrinkled between two eyebrows
4.necessary: according to brows' action may be upper lid is lowered and tense
5.necessary : lips are either pressed firmly together with corners straight or down or open
6. nostrils widened
Ending: outward raising eyebrows and mouth expansion

Table A.5: Facial Expression Display: Fear.

Facial Expression Display 6: Fear
Beginning: sight expansion and lowering of mouth and raising inner parts of brows
1. sufficient: brows raised and drawn together
2. necessary: forehead wrinkles go to the center
3. sufficient : upper eyelid is raised and lower eyelid is drawn up
4. necessary: lips are slightly tense or stretched and drawn back
5. necessary: mouth is open
Ending: sight contraction and raising of mouth and lowering inner parts of brows

The following Figures show the definition of the 20 flow regions which are related to a subset of facial muscles to be suitable to automatically detect the six well-known basic emotions.

Table A.6: Descriptions of Facial Transient Feature Regions in Eyes.

Region 1 VP1	Left eye inner corner, stable point
Region 2 VP2	Right eye inner corner, stable point
Region 3 VP3	Left eye outer corner, stable point
Region 4 VP4	Right eye outer corner, stable point
Region 5 VP5	Top of the left eye, non-stable
Region 6 VP6	Top of the right eye, non-stable
Region 7 VP7	Bottom of the left eye, non-stable
Region 8 VP8	Bottom of the right eye, non-stable

Table A.7: Descriptions of Facial Transient Feature Regions in Brows.

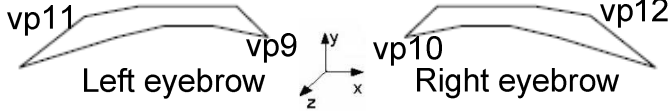

<p>Region 9 VP9 Left eyebrow inner corner, non-stable</p> <p>Region 10 VP10 Right eyebrow inner corner, non-stable</p> <p>Region 11 VP11 Left eyebrow outer corner, non-stable</p> <p>Region 12 VP12 Right eyebrow outer corner, non-stable</p>

Table A.8: Descriptions of Facial Transient Feature Regions in Nose.

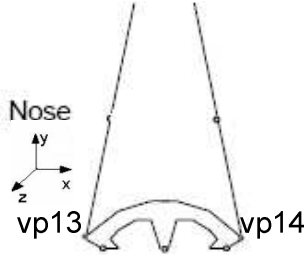
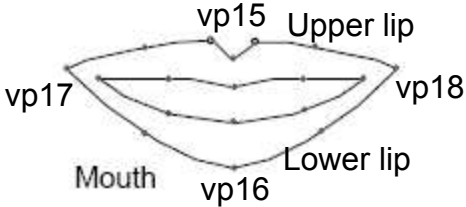
	<p>Region 13 VP13 Left nostril centre, non-stable</p> <p>Region 14 VP14 Right nostril centre, non-stable</p>
--	--




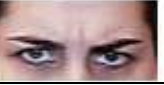







Table A.9: Descriptions of Facial Transient Feature Regions in Mouth.






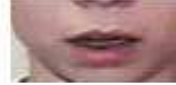


<p>Region 15 VP15 Top of the upper lip, non-stable</p> <p>Region 16 VP16 Bottom of the lower lip, non-stable</p> <p>Region 17 VP17 Left corner of the mouth, non-stable</p> <p>Region 18 VP18 Right corner of the mouth, non-stable</p>

Appendices B

Optical Flow-based Face Feature Representation for AUs Recognition

Table B.1 Face Feature Representation for AUs Recognition.

AU	FACS description	Example images	Permanent Features (compared to natural expression)
1	Raised inner brows		In inner brow region Intensity increased & Angle decreased
2	Raised outer brow		In outer brow region Intensity increased & Angle decreased
1+2	Raised eyebrows		Intensity in furrow region & Intensity in brow region & Angle decreased
4	Lowered brows or Frowned brows		Intensity in furrow region & Intensity in brow region & Angle increased
5	Raised upper lid		In up-lids region Intensity increased & Angle decreased
6	Raised cheek		In cheek region Intensity increased & Angle increased
7	Raised lower lid		In lower lid region Intensity increased & Angle decreased
9	Wrinkled nose		In furrow region Intensity increased & Angle decreased
10	Raised upper lip		In up-lip Intensity increased & Angle decreased
12	Mouth corners pulled up		In right lip corner Intensity Increased & Angle Increased In left lip corner Intensity Increased & Angle Decreased
15	Mouth corner pulled downwards		In right lip corner Intensity Increased & Angle Decreased In left lip corner Intensity Increased & Angle Increased

17	Raised chin		In Chin region Intensity Increased & Angle Decreased
20	Mouth stretched		In Mouth Region Intensity Increased & Angle Decreased
23	Lips tightened but not pressed		In Mouth Region Intensity Increased & Angle Increased In lip corner Region Intensity Increased & Angle Decreased
24	Lips pressed together		In Mouth Region Intensity Increased & Angle Increased much more In lip corner Region Intensity Increased & Angle Decreased much more
25	Lips parted		In Mouth Region Intensity Increased & Angle Decreased In lip corner Region Intensity Increased & Angle Increased
26	Jaw dropped		In Mouth Region Intensity Increased & Angle Decreased In lip corner Region Intensity Increased & Angle Increased
27	Mouth stretched		In Mouth Region Intensity Increased much & Angle Decreased much In lip corner Region Intensity & Angle Increased much

Appendices C

DBN Modeling Software & Toolbox

DBN modeling tools-GeNIe

GeNIe stands for Graphical Network Interface which is a graphical user interface. It is implemented in C++ and makes heavy use of the Microsoft Foundation Classes. Its emphasis is on accessibility and friendliness of the user interface, making creation of decision theoretic models intuitive using a graphical click-and-drop interface approach. It is a versatile and userfriendly environment for building graphical decision models and performing diagnosis.

- **Modeling with a DBN**

In this section we will discuss the arrangement of the developed Bayesian network. First, I will provide a brief review of the approach to the network's construction. Furthermore, I will try to illustrate and explain the actual layout of the network.

First of all, according to the GeNIe tutorial document I present the main steps of creating a general DBN in Graphical user interface (GeNIe) and give some example Figures.

Haven mentioned in theory part that in the DBN formalism, the node has four temporal types which are anchor, co temporal, plate, and terminal. But in a static network which means in BN, the types of node are different because the concept of temporal types does not exist in BN. One of the most differences between static network and dynamic network is about the concept of temporal. So in the following section I will present the approach in creating our DBN from six parts which are temporal types, temporal arcs, temporal definition, temporal evidence, temporal posterior beliefs and unrolling.

- **Temporal types**

In the Graphical user interface design for dynamic networks or we can call temporal networks, we should connect the temporal types with the GeNIe already existed rather than to change the already existed. The following way is that how to add temporal relations in a static network. First a new network is static by default as shown in Figure C.1

Next step is to click on Network -> Enable Temporal Plate in the menu bar as shown in Figure C.2. So far we get the result that in the network length/angle being divided into four parts which are temporals, initial conditions, temporal plate and terminal conditions, according to the four temporal types.

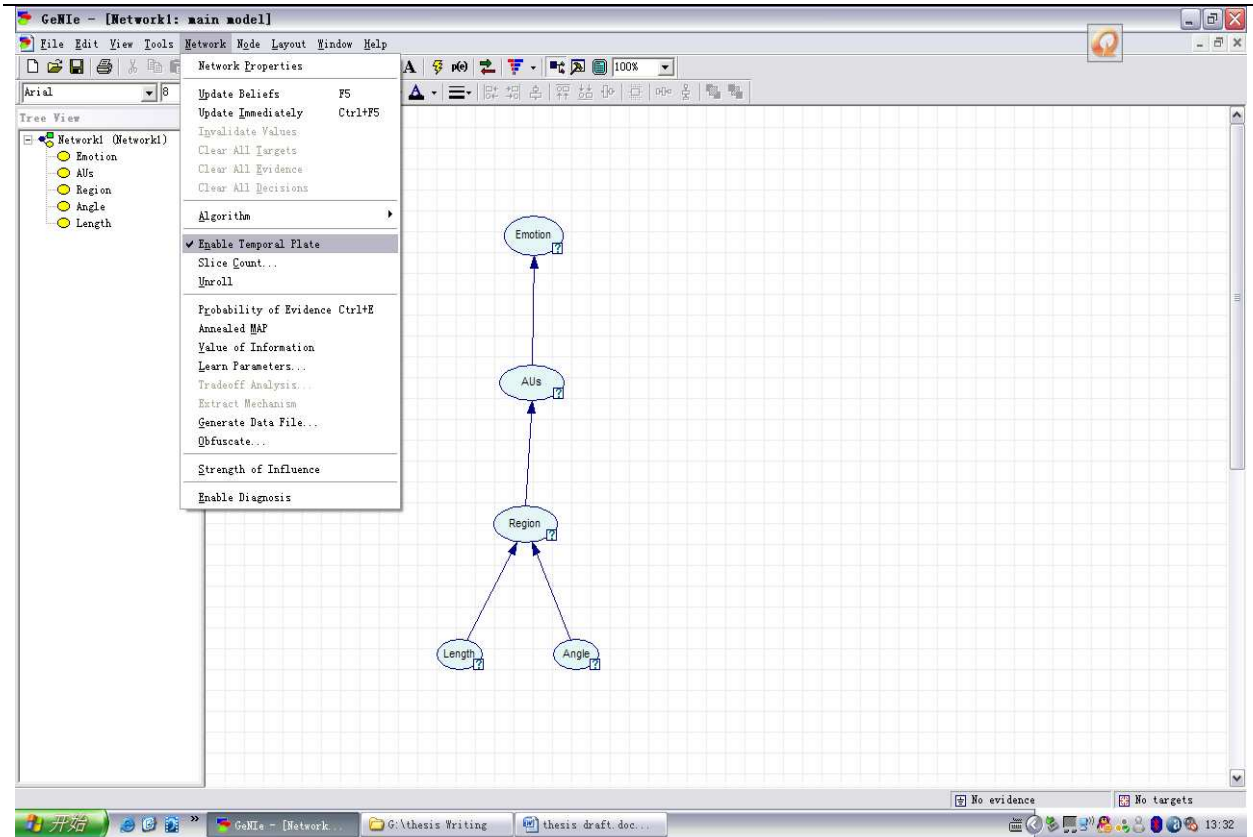


Figure C.1: Temporal options in the menu bar of GeNIe under Network.

[GeNIe & SMILE]

1. **Contemporals:** this is the part of the network length/angle where the static nodes are stored by default.
2. **Initial conditions:** this is the part of the network length/angle where the anchor nodes are stored.
3. **Temporal plate:** this is the part of the network where the nodes in the temporal plate are stored. Nodes in the temporal plate are the only nodes that are allowed to have temporal arcs. This length/angle also shows the number of time-slices for which inference is performed.
4. **Terminal conditions:** This is the part of the network length/angle where the terminal nodes are stored.

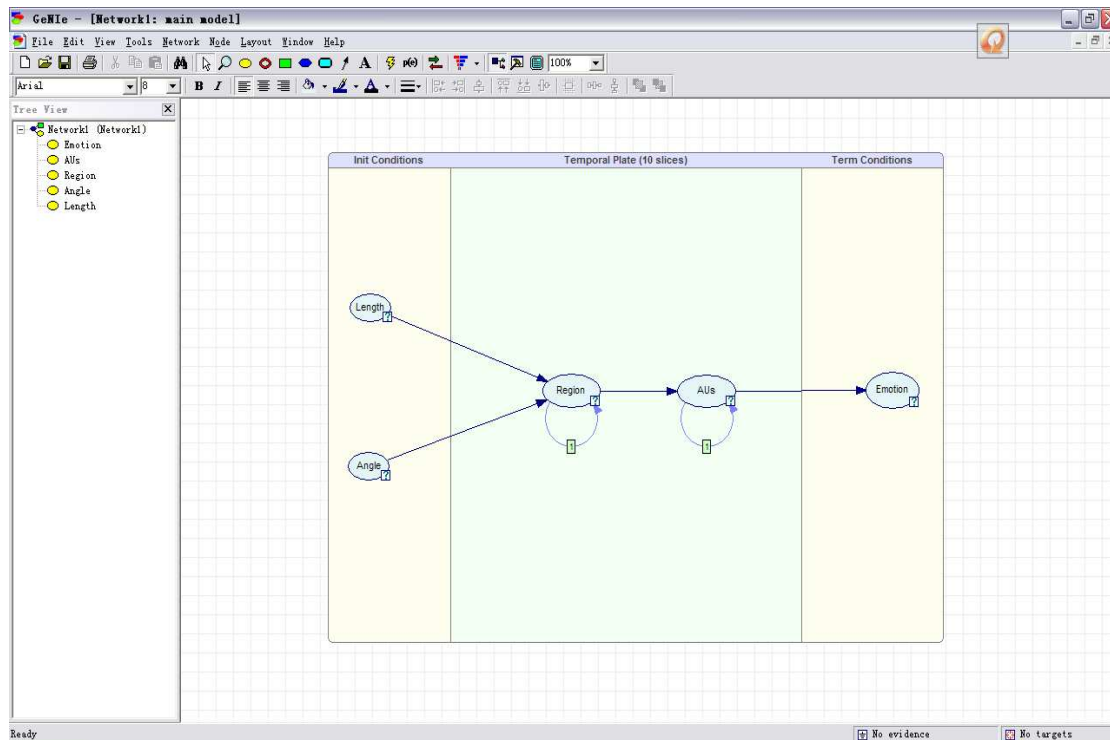


Figure C.2: General overview of the network length/angle environment in GeNIe.

The boundaries of the different Length/Angles can be changed by dragging them. And when the network is unrolled explicitly these boundaries are used to layout the network. Moreover, when the network modeler wants to change the temporal he just need drag them to one of the three other Length/Angles. A general overview of the network length/angle environment is shown in Figure C.2.

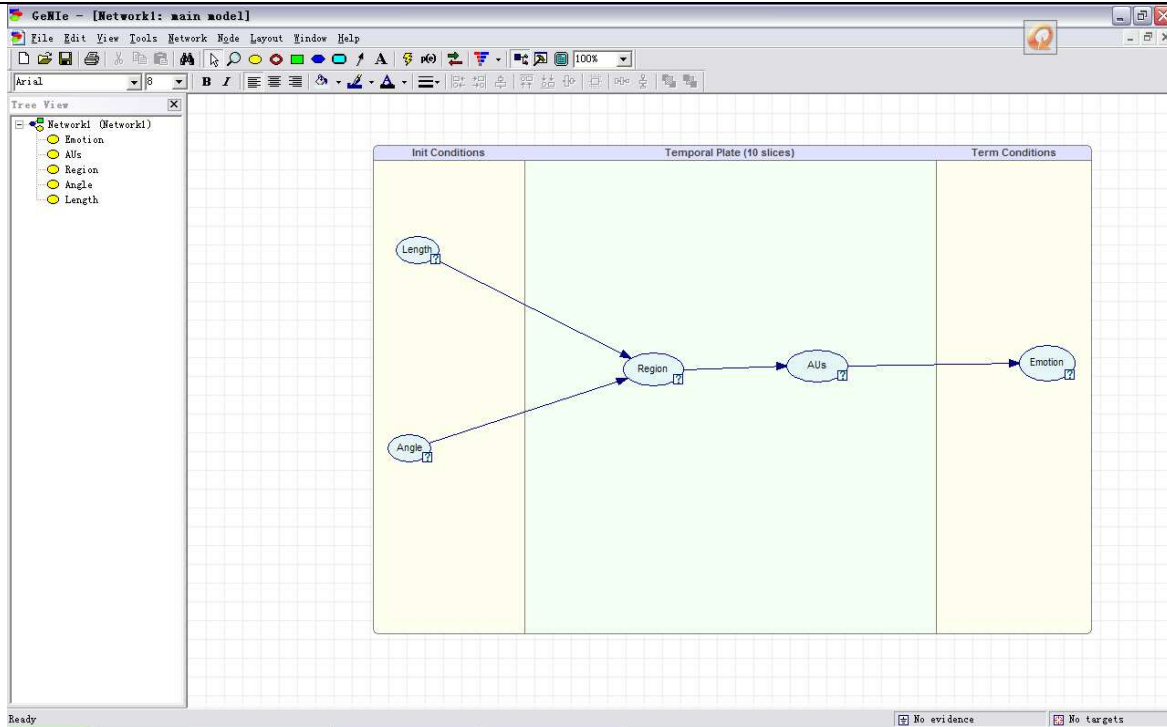


Figure C.3: demonstrates the addition of a temporal arc to a network.

Temporal arcs : after finished dragging a node to the temporal plate, temporal arcs need to be added in the network. Generally speaking, a temporal arc is composed of a parent, a child and a temporal order. A temporal arc can be added by clicking on the arc button and drawing an arc from a parent to a child². When the arc has been drawn in the network the mouse button need to be released, and then a context menu appears to enable the user to set the temporal order of the arc. Every temporal arc has a small label with its temporal order. Figure C.3 demonstrates the addition of a temporal arc to a network.

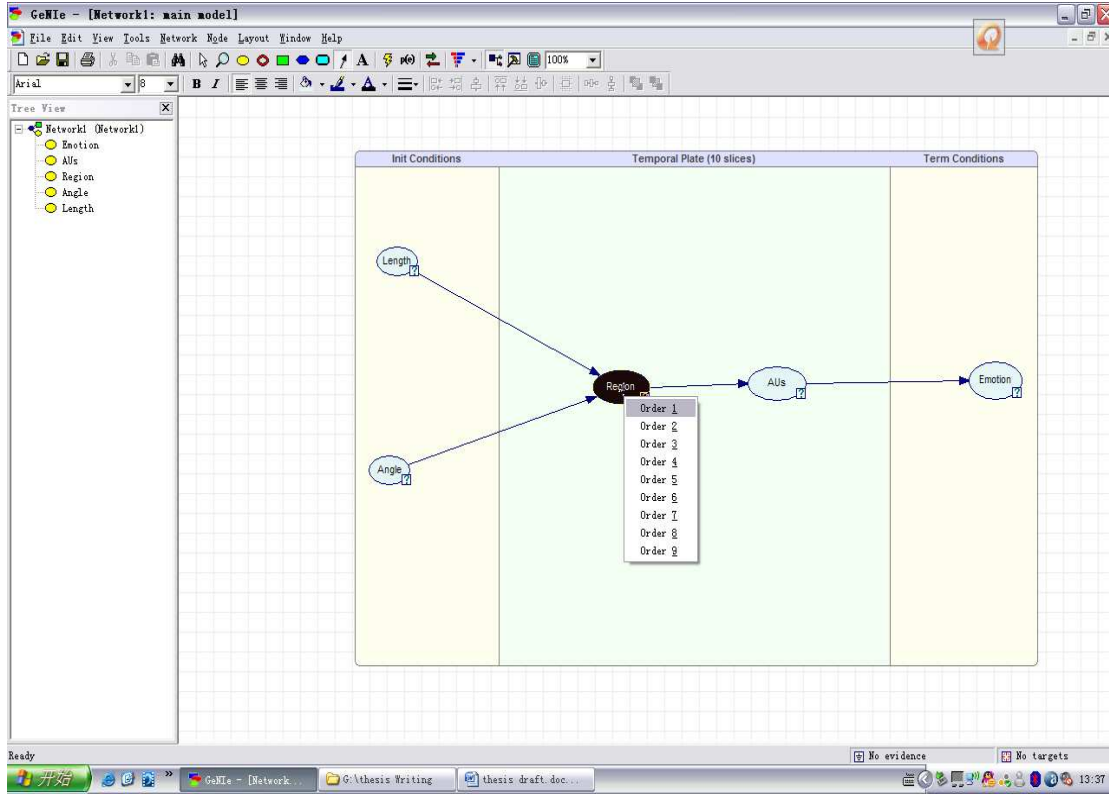


Figure C.4: Deciding a target node.

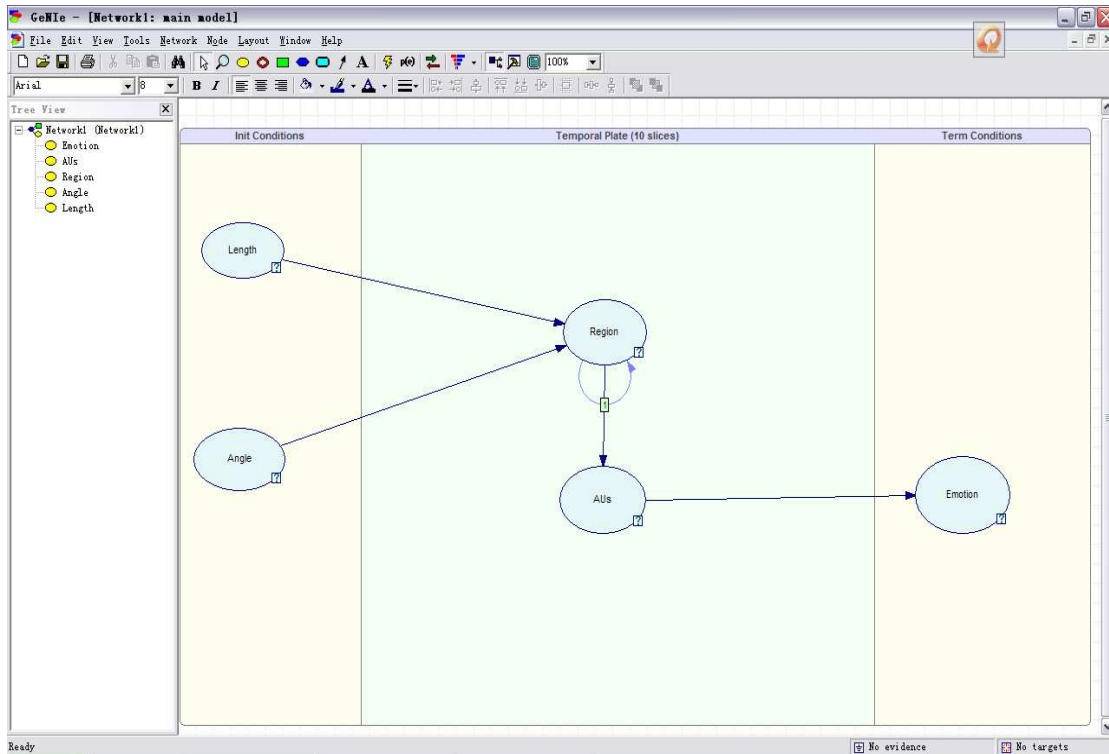


Figure C.5: Adding a temporal arcs in GeNIe.

- **Temporal definition**

After added the temporal arcs for each node a CPD should be added for every node in the temporal plate because every node needs a CPD for every incoming temporal arc with a different temporal order. The CPD can be set by double-clicking on the node in the network length/angle and go to the definition tab to set the temporal CPDs just like setting the CPD for ordinary nodes. After added an incoming temporal arcs for each node, the appropriate temporal CPD from a list can be selected by the network modeler and its parameters can be set by the network modeler as well. Figure C.5 demonstrates this process.

- **Temporal evidence**

After a temporal network finished it can be used for decision support. But before the observations or evidence can be added to the temporal network, the number of time-slices of the temporal network needs to be set by clicking on Network → Slice Count. The number of time-slices denotes the time-period the decision maker is interested in. After setting the number of time-slices, the evidence can be added to nodes in the temporal network by right-clicking on a node and selecting Evidence from the context menu. If the nodes are non-temporal, the evidence can be set directly by the decision. But for temporal nodes, the evidence can be added for every time-slice in which a form will appear. Figure C.6 demonstrates the addition of evidence to a temporal node.

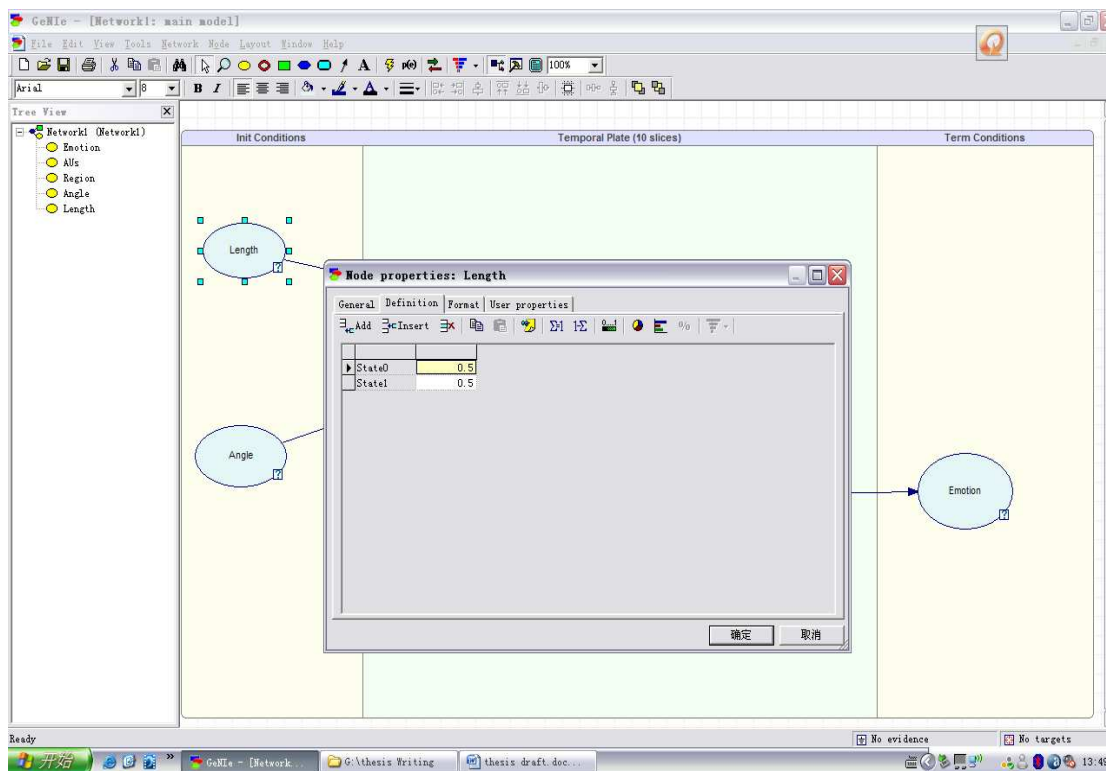


Figure C.6: Setting the probabilities for values in GeNIe.

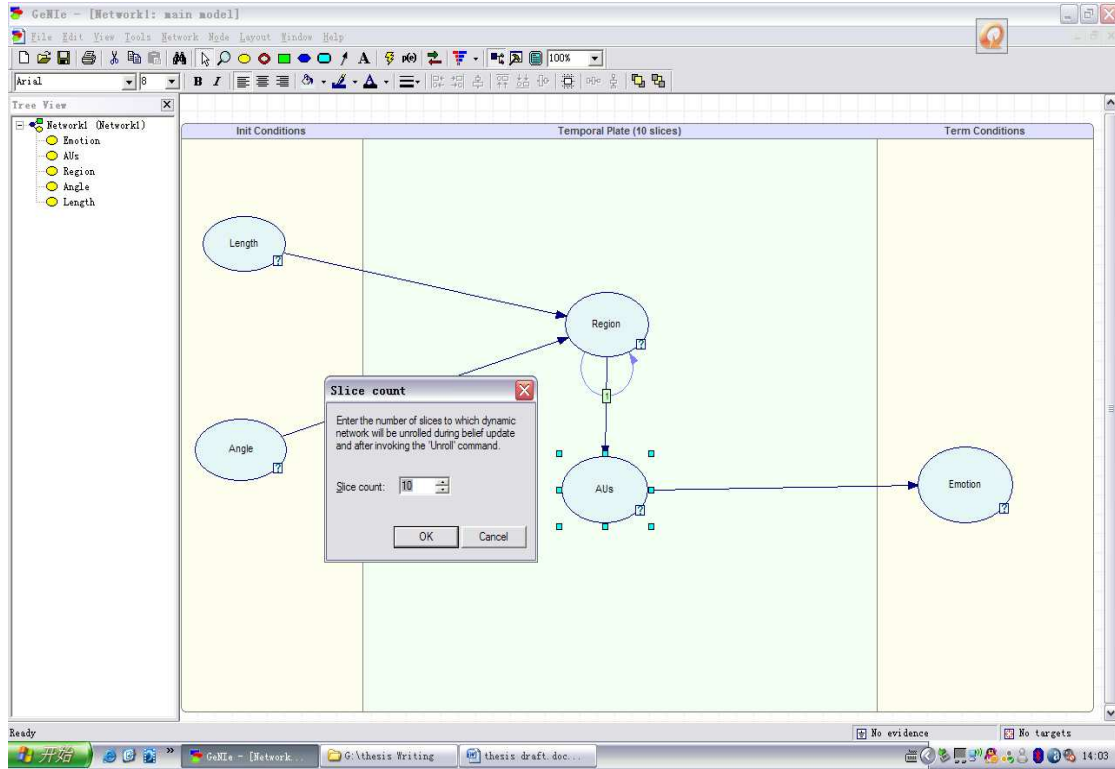


Figure C.7: Setting the probabilities for values in GeNIe.

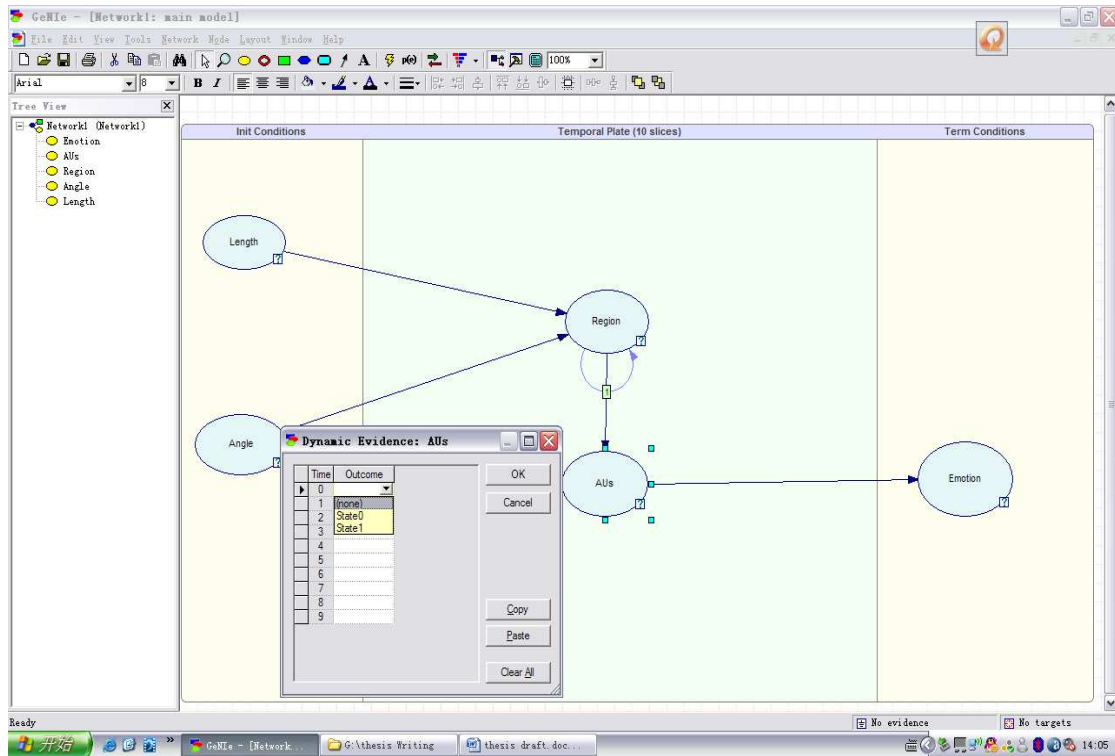


Figure C.8: Setting temporal evidence in GeNIe.

● *Temporal inference*

In fact, there is no difference between calling inference on a temporal network and calling inference on a non-temporal network from a GeNIe perspective. In inference progress the inference algorithm should be selected and set by the decision maker by clicking on Network → Algorithm. . . and selecting the appropriate algorithm. The algorithm is called by right-clicking on the network length/angle and selecting the button Update Beliefs from the context menu or by pressing F5 in keyboard.

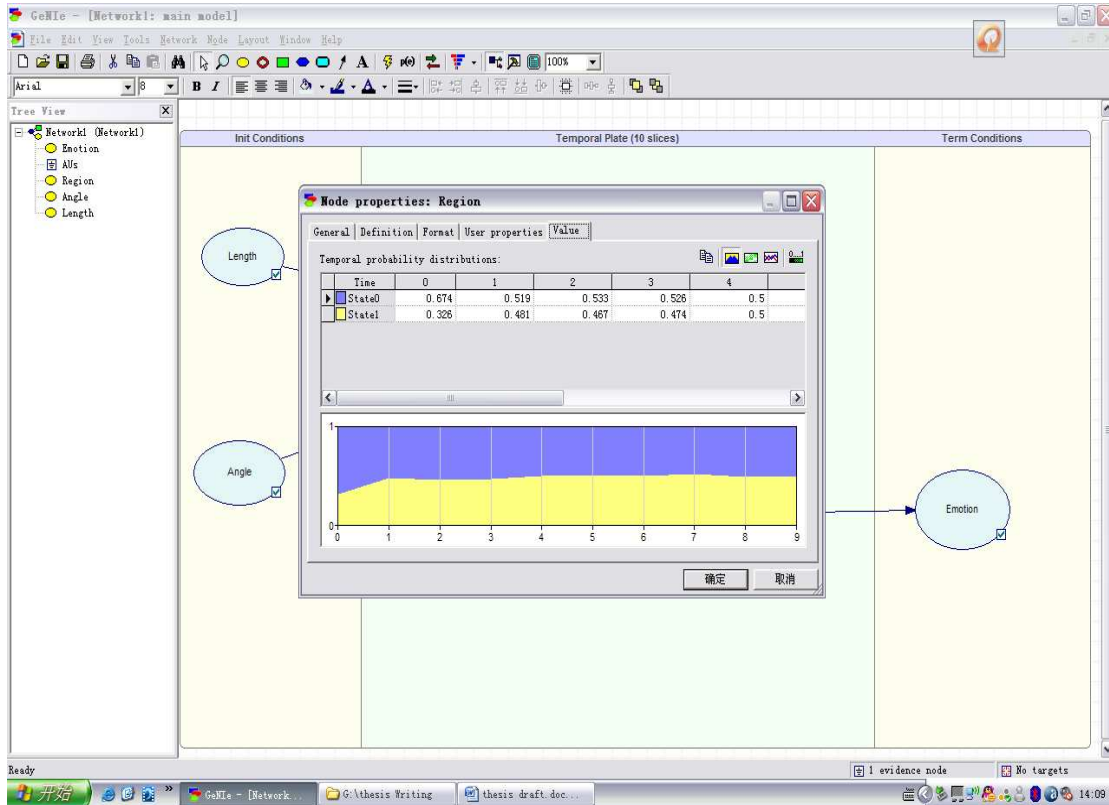


Figure C.9: After training the network by algorithm.

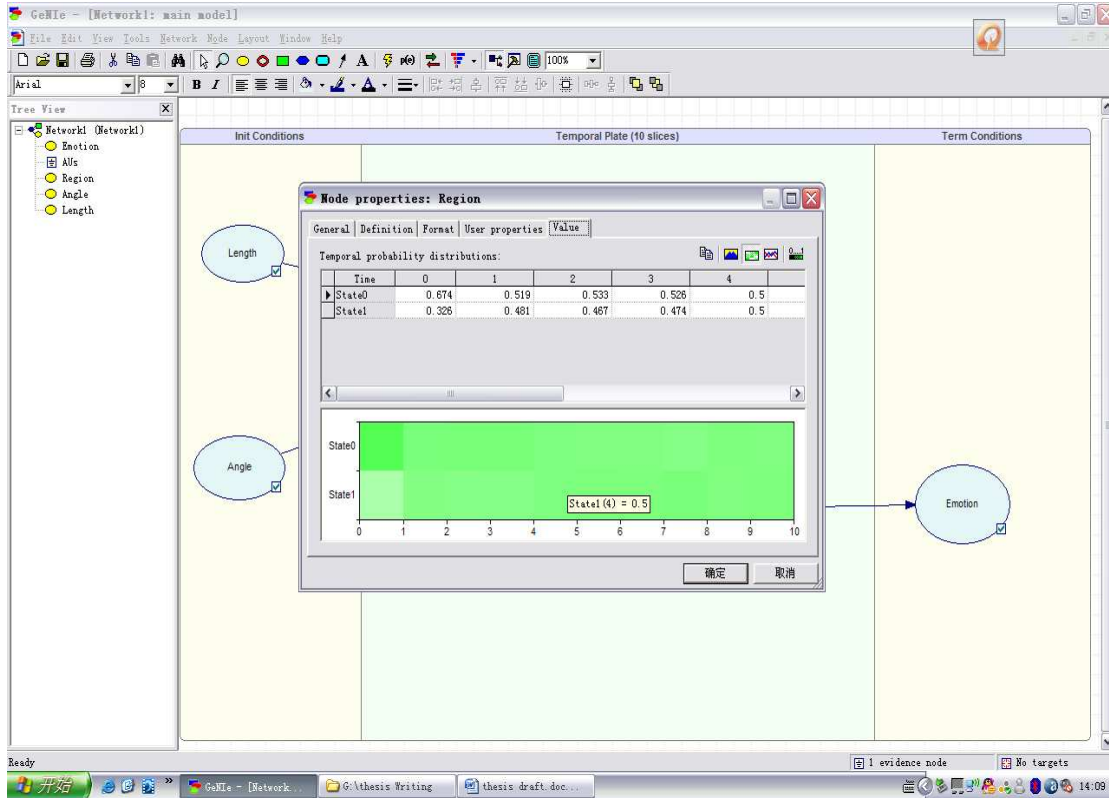


Figure C.10: Showing different representations of the temporal posterior beliefs.

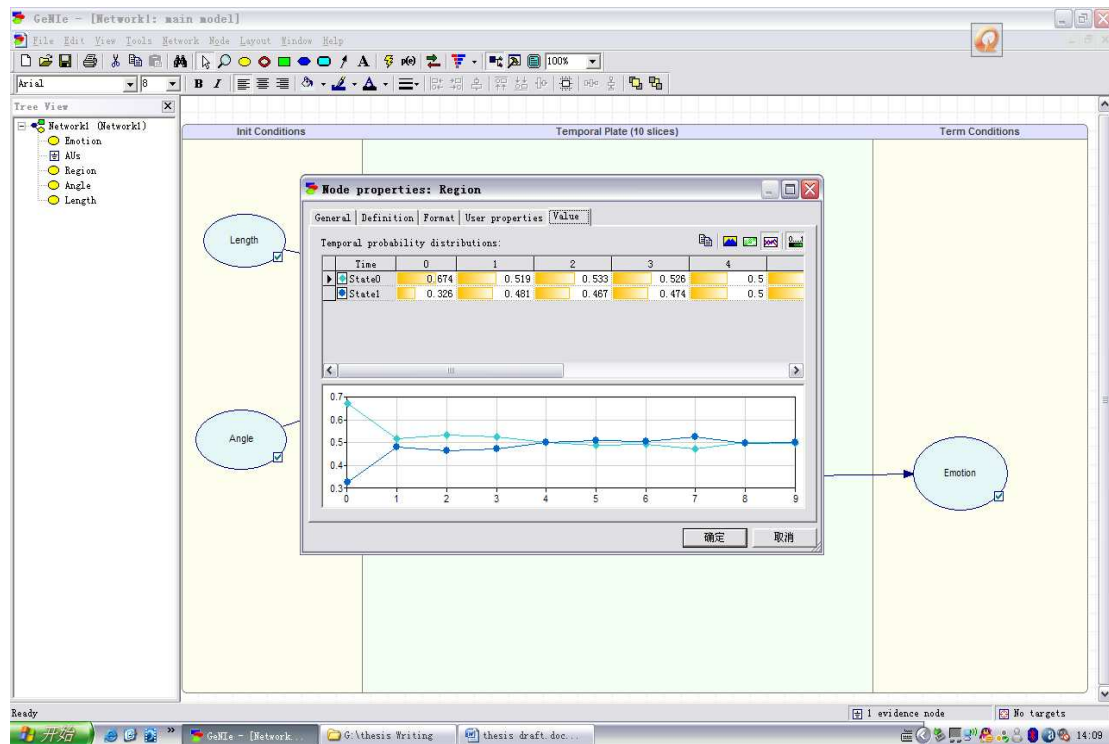


Figure C.11: Showing different representations of the temporal posterior beliefs in GeNIe.

- ***Temporal posterior beliefs***

After inference is called, the temporal network has its beliefs updated. It is very important for the decision maker that these updated beliefs can be presented clearly. The updated beliefs for a temporal node can be obtained by double-clicking on the node and selecting the Value tab. This tab contains the updated beliefs for all time-slices of that node, in addition the updated beliefs can be shown not only in floating point numbers, but also as: contour plot, a length/angle chart, and time-series plot. Both the length/angle chart and time-series plot can be found in other packages as well, but the contour plot is a representation of temporal beliefs that is a unique feature of GeNIe to the best of our knowledge. The contour plot represents a three dimensional space with the dimensions: time (x-axis), state (y-axis), and probability of an outcome at every point in time (the color). It is possible for the decision maker to cut-off the color range to get more insight into the development and the uncertainty of the temporal process. Generally speaking, the representation of contour plot is especially useful for temporal nodes that have many states; in contrast, the other two representations are more useful for temporal nodes with not too many states. Figure c10, 11 shows the different representations of the temporal posterior beliefs.

- ***Unrolling***

It can be useful for debugging purposes to explicitly unroll a temporal network for a given number of time-slices. GeNIe provides this possibility through the Unroll option in Network menu. When clicking this option, GeNIe opens a new network which is the temporal network has been unrolled according to the given number of time-slices. To locate a node in the temporal network from the unrolled network that is also possible, just needs right-click on the node in the unrolled network and select Locate Original in DBN from the context-menu. The unrolled network that is a result from unrolling the temporal network is cleared from any temporal information whatsoever. It can be edited, saved and restored just like any other static network. Figure C. 15 shows the unrolled representation of a temporal network.

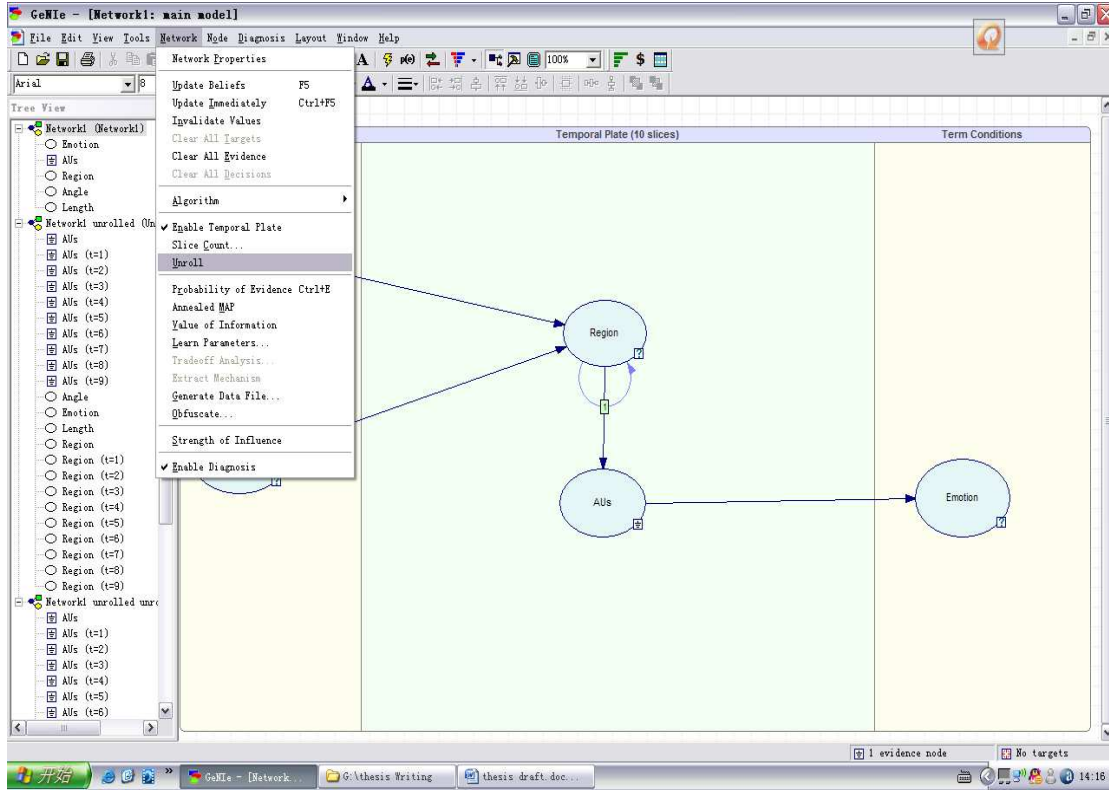


Figure C.12: Unrolling a temporal network and locating a node in GeNIe.

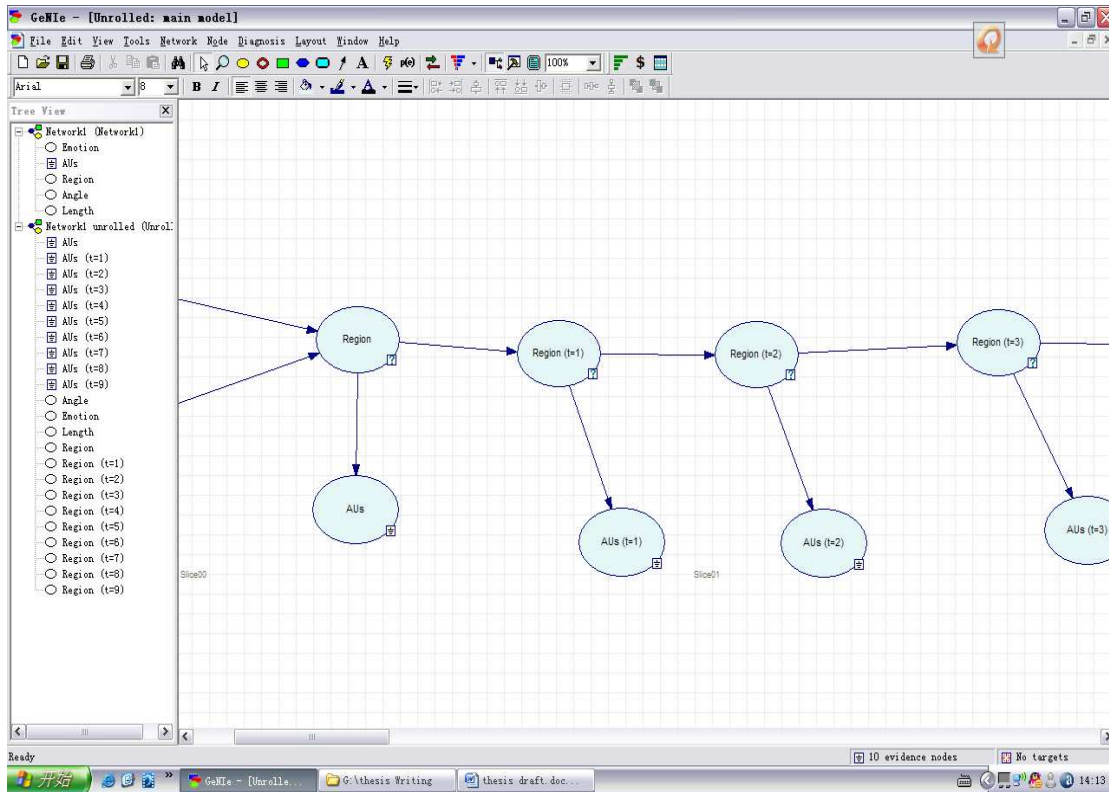


Figure C.13: Unrolling a temporal network and locating a node in GeNIe.

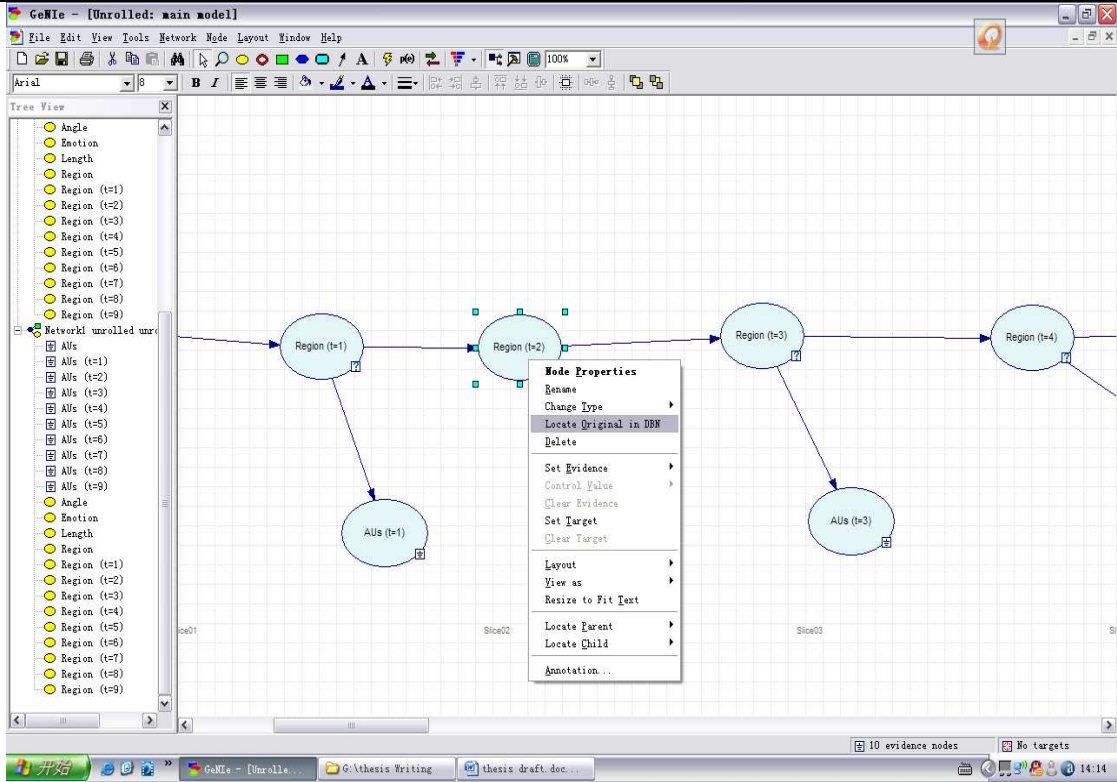


Figure C.14: Unrolling a temporal network and locating a node in GeNIe.

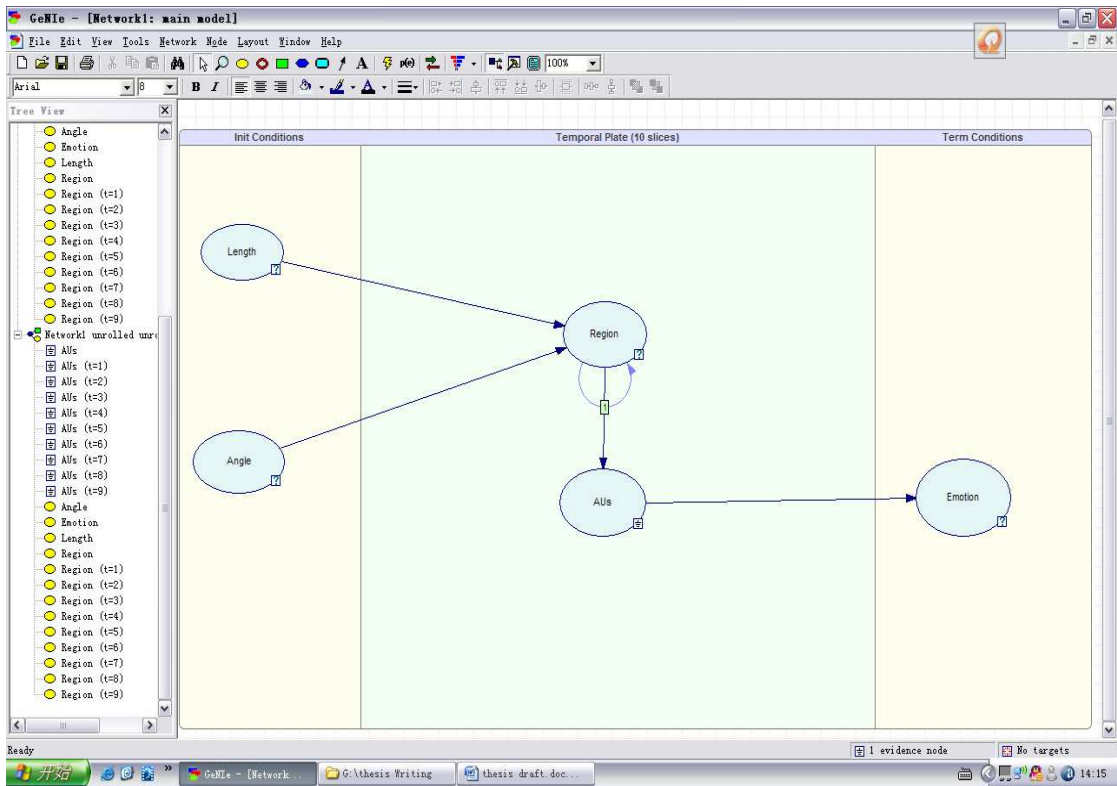


Figure C.15: Unrolling a temporal network and locating a node in GeNIe.

Notices in unrolling:

- To avoid an explosion of the model complexity, parameters are tied such that the transition probabilities between time slices are the same.
- Contours within a time slice are not allowed.
- A limitation was set on the maximum number of contours converging on a single node.

Appendices D

Annotation of Cohn-Kanade database

The images used in the labeling procedure come from the Cohn-Kanade Database. The database consists of expression sequences of subjects, starting from a neutral expression and ending most of the time in the peak of the facial expression. Subjects are university students enrolled in introductory psychology classes. They ranged in age from 18 to 30 years. Subjects were instructed by an experimenter to perform a series of 23 facial displays. Six of the displays were based on descriptions of prototypic emotions (i.e., happiness, anger, fear, disgust, sadness and surprise). There are 104 subjects in the database and only 10 of them gave the consent for publications.

Annotation Process

The annotation process consists in associate AUs expression label to each of the images from Cohn-Kanade database. A simple and intuitive interface has been designed in order to facilitate the annotation process:

- The first time we have to input *Image Dir*, *Image Type* and *Saving Location*. By clicking on *PreProcess Dir* we will see the image will be labeled.
- By clicking on Load Last Record we will save the information has been filled according to the saving location. As shown in the following Figure D.1.

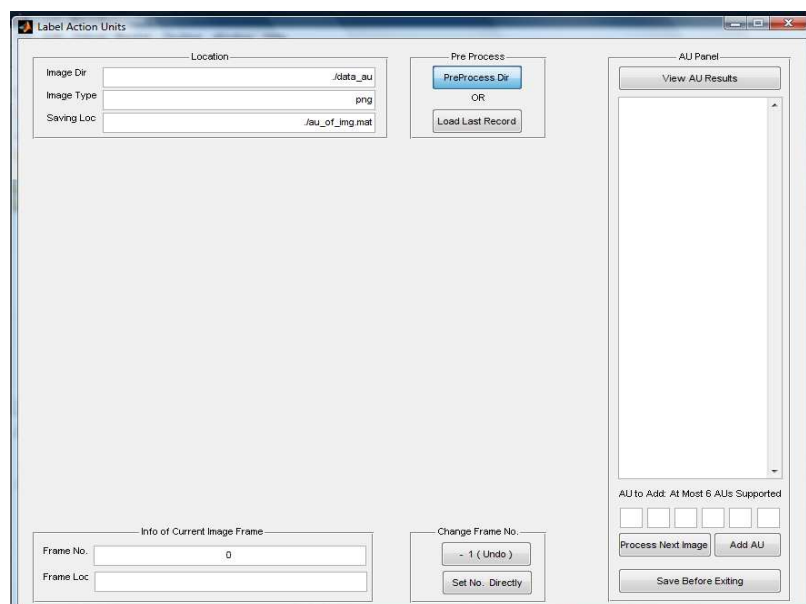


Figure D.1: Interface of inputting image.

- By clicking on **Process Next Image** we will pass to the next unlabelled frame. For each image should be labeled by the additions of some AU.
- By inputting the AUs number in those regions below **AU to add at Most 6 AUs Supported** we will start the labeling procedure for the unlabelled images. The labeling interface is shown in the following Figure D.2.

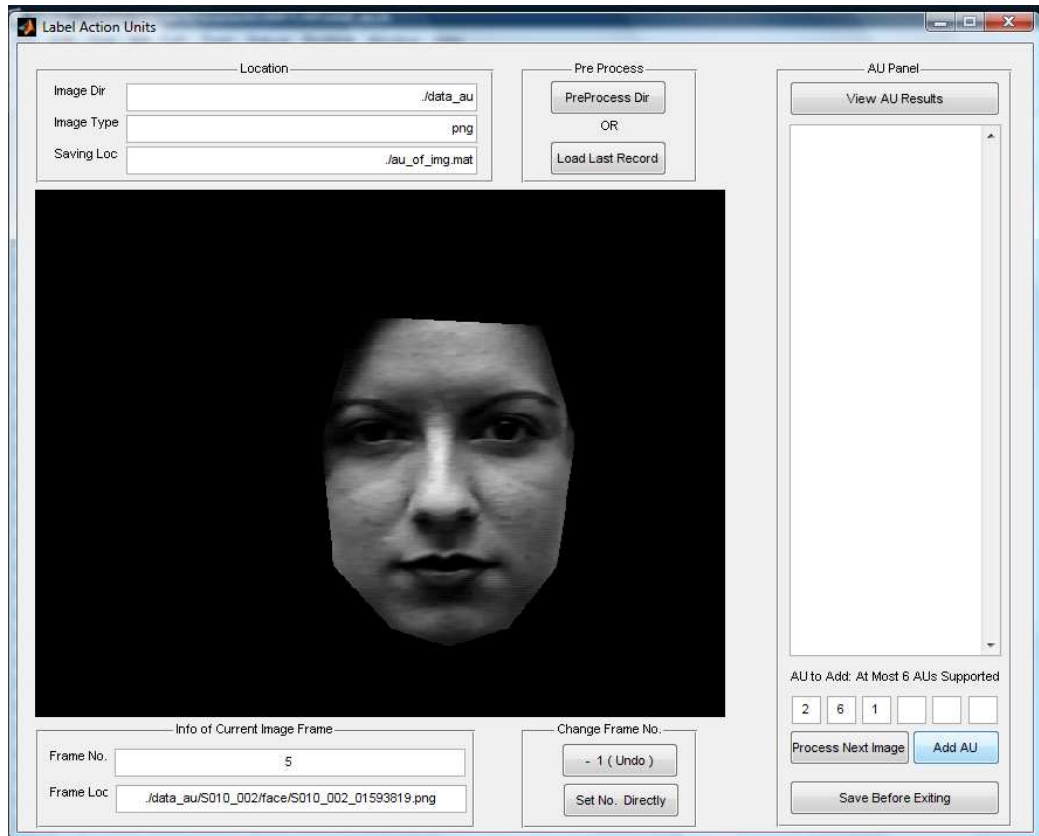


Figure D.2: Inputting the AUs number.

- By clicking on **Add AU** we will save those AUs which are used to label each images. As shown in the following Figure D.3.

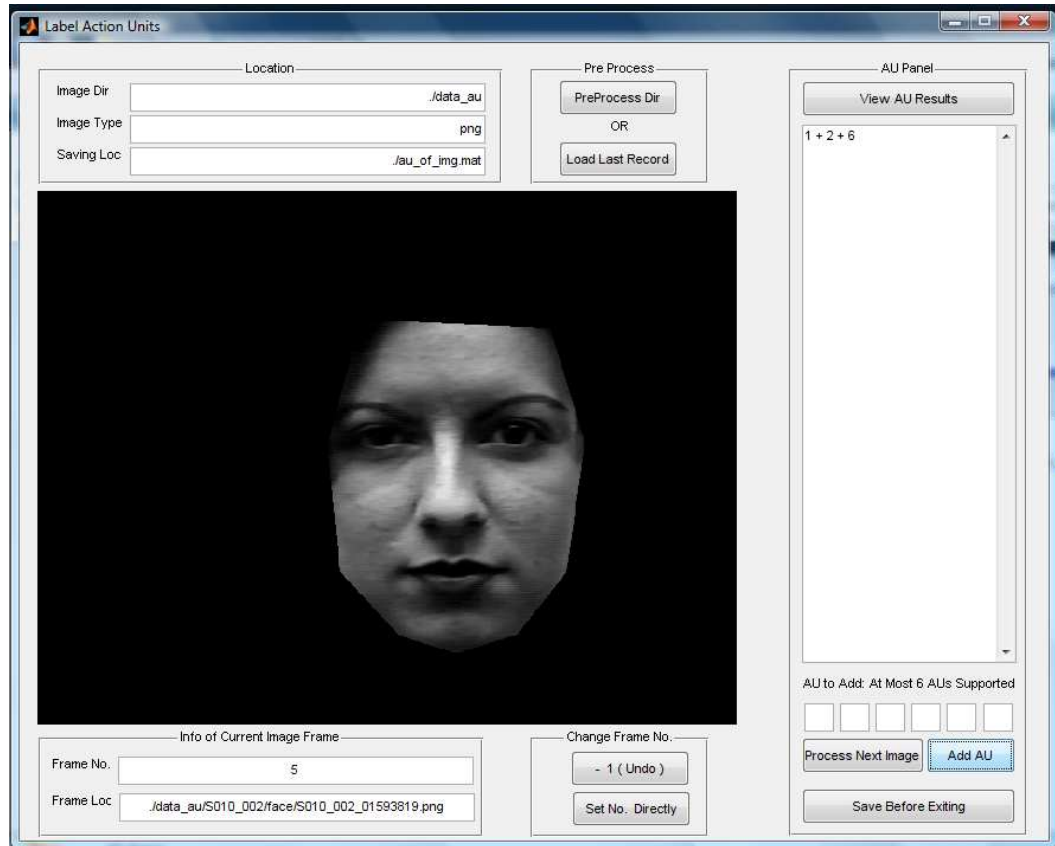


Figure D.3: Saving the labeled images with AUs number.

- By clicking on **Save Before Exiting** we will save the annotation of each images. As shown in the following Figure.
- By clicking on **View AU Results** we can check the labeled frames with the expression AUs so that we can clearly know which AUs are combined to express each frame.

Appendices E

ROI Selection & Optical Flows Detection from ROI Matlab Code

```

function varargout = calc_roi_mask(varargin)
% CALC_ROI_MASK M-file for calc_roi_mask.fig
%     CALC_ROI_MASK, by itself, creates a new CALC_ROI_MASK or raises the
existing
%     singleton*.
%
%     H = CALC_ROI_MASK returns the handle to a new CALC_ROI_MASK or the
handle to
%     the existing singleton*.
%
%     CALC_ROI_MASK('CALLBACK',hObject,eventData,handles,...) calls the
local
%     function named CALLBACK in CALC_ROI_MASK.M with the given input
arguments.
%
%     CALC_ROI_MASK('Property','Value',...) creates a new CALC_ROI_MASK or
raises the
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before calc_roi_mask_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property application
%     stop. All inputs are passed to calc_roi_mask_OpeningFcn via varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help calc_roi_mask

% Last Modified by GUIDE v2.5 30-Dec-2008 18:26:15

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',  gui_Singleton, ...
                  'gui_OpeningFcn', @calc_roi_mask_OpeningFcn, ...
                  'gui_OutputFcn',  @calc_roi_mask_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

```

```

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before calc_roi_mask is made visible.
function calc_roi_mask_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to Figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to calc_roi_mask (see VARARGIN)

% Choose default command line output for calc_roi_mask
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes calc_roi_mask wait for user response (see UIRESUME)
% uiwait(handles.Figure1);

% --- Outputs from this function are returned to the command line.
function varargout = calc_roi_mask_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to Figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% % get image directory here, save in handles.img_dir
% % default dir is './data/'
function edit_image_dir_Callback(hObject, eventdata, handles)
% hObject    handle to edit_image_dir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_image_dir as text
%        str2double(get(hObject,'String')) returns contents of edit_image_dir
%        as a double

% % get image dir name where to process images
img_dir = get(hObject,'String');

```

```
% % default image dir name
if isempty(img_dir); img_dir = './data/'; end;
% % dir name should end with '/'
if img_dir(end) ~= '/' && img_dir(end) ~= '\'; img_dir = [img_dir, '/'] ; end;
% % record this dir name in handles
handles.img_dir = img_dir;
% % save handles
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function edit_image_dir_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_image_dir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% % just show the name of the file under processing here
function edit_current_file_Callback(hObject, eventdata, handles)
% hObject    handle to edit_current_file (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_current_file as text
%         str2double(get(hObject,'String')) returns contents of
edit_current_file as a double

% --- Executes during object creation, after setting all properties.
function edit_current_file_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_current_file (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% % get location for saving processed data
function edit_save_loc_Callback(hObject, eventdata, handles)
% hObject    handle to edit_save_loc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```

% Hints: get(hObject,'String') returns contents of edit_save_loc as text
%         str2double(get(hObject,'String')) returns contents of edit_save_loc
as a double

% % record location for saving processed data
save_loc = get(hObject,'String');
if save_loc(end) ~= '/' && save_loc(end) ~= '\'; save_loc = [save_loc, '/'] ;
end;
if ~exist(save_loc,'dir'); mkdir(save_loc); end;
handles.compass_loc = [save_loc,'compass/'];
if ~exist(handles.compass_loc,'dir'); mkdir(handles.compass_loc); end;
handles.histogram_loc = [save_loc,'histogram/'];
if ~exist(handles.histogram_loc,'dir'); mkdir(handles.histogram_loc); end;
handles.save_loc = save_loc;
handles.mtrx_loc = [save_loc,'roi_mask.mat'];
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function edit_save_loc_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_save_loc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% % get image type
function edit_image_type_Callback(hObject, eventdata, handles)
% hObject    handle to edit_image_type (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_image_type as text
%         str2double(get(hObject,'String')) returns contents of
edit_image_type as a double

% % record image type
handles.img_type = get(hObject,'String');
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function edit_image_type_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_image_type (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.

```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% % pre-process data
% % locations of all the image files are recorded in handles.roi_mask
% --- Executes on button press in pushbutton_pre_process_dir.
function pushbutton_pre_process_dir_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_pre_process_dir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% % init info for pre-processing data
if ~isfield(handles,'img_dir'); handles.img_dir = './data/'; end;
if ~isfield(handles,'img_type'); handles.img_type = 'png'; end;
% % create roi_mask for image files in the dir
handles.roi_mask = cell(0,1);
handles.roi_mask = roi_mask_create_struct(handles.roi_mask,handles.img_dir,handles.img_type);
% % init file being processed
handles.current_file_index = 0;
% % init rects' positions: [xmin,ymin,width,height]
% % rects: 18+3
handles.vpts_rect = [...
%     340,215,20,20;
%     390,215,20,20;
%     265,220,20,20;
%     470,220,20,20;
%     305,200,20,20;
%     435,200,20,20;
%     305,230,20,20;
%     430,225,20,20;
%
%     340,195,20,20;
%     390,195,20,20;
%     275,195,20,20;
%     465,185,20,20;
%
%     340,290,20,20;
%     405,290,20,20;
%
%     375,325,20,20;
%     375,350,20,20;
%     325,335,20,20;
%     425,335,20,20;
%
%     365,160,20,20;
%     290,335,20,20;
%     460,335,20,20;
%     375,380,20,20;
% ];
    340    219    23    25;
```

```

385 215 25 25;
266 226 25 27;
467 220 23 27;
304 200 27 20;
429 200 26 20;
305 230 26 21;
430 225 26 20;
340 193 23 22;
385 193 25 22;
274 195 23 23;
465 189 23 25;
340 287 28 27;
393 283 29 27;
367 327 37 20;
367 351 38 20;
325 335 30 27;
415 331 28 24;
337 142 90 37;
278 291 41 51;
448 289 35 50;
363 385 53 29;
];
% % save them in handles
guidata(hObject,handles);

% % show and manually move rectangles
% --- Executes on button press in pushbutton_process_next_image.
function pushbutton_process_next_image_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton_process_next_image (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% % update file index to process
handles.current_file_index = handles.current_file_index+1;
% % all the files processed?
if handles.current_file_index >= numel(handles.roi_mask);
    % % show the info in the interface
    set(handles.edit_current_file,'String','All images have been
processed!');
    drawnow;
else
    % % show the location in the interface
    img_loc = handles.roi_mask{handles.current_file_index}.loc;
    set(handles.edit_current_file,'String',img_loc);
    drawnow;
    % % read image file
    img = imread(img_loc);
    % % show the image in the interface
    axes(handles.axes_show_current_image);
    imshow(img);
    % % draw rects in the image
    % % record the handles of these rects in handles.hrect
    for i = 1:size(handles.vpts_rect,1);

```



```

        handles.hrect{i,1}
imrect(handles.axes_show_current_image,handles.vpts_rect(i,:));
    end;
end;
guidata(hObject,handles);

% % unprocess this image
% --- Executes on button press in pushbutton_unprocess_this_image.
function pushbutton_unprocess_this_image_Callback(hObject, eventdata,
handles)
% hObject handle to pushbutton_unprocess_this_image (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% % just set the index of file to the former one
% % data is actually not affected
handles.current_file_index = handles.current_file_index - 1;
guidata(hObject,handles);

% % record the rects that have been manually adjusted
% % record the flow vectors within these rects
% --- Executes on button press in pushbutton_record_the_rect.
function pushbutton_record_the_rect_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton_record_the_rect (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% % record the adjusted rects
for k = 1:numel(handles.hrect);
    handles.vpts_rect(k,:) = getPosition(handles.hrect{k});
end;
% % also record the rects for every image file
dex = handles.current_file_index;
handles.roi_mask{dex}.rect = handles.vpts_rect;
% % load the flow vectors
pctdata = roi_mask_load_pct_data(handles.roi_mask{dex}.pctloc);
% % record those vectors within the rects
tmp_vec = roi_mask_roi_flow_vectors(pctdata,handles.vpts_rect);
handles.roi_mask{dex}.roiflowvectors = tmp_vec;

type = handles.img_type;
fname = handles.roi_mask{dex}.name;
% % compass and save
if ~isfield(handles,'save_loc');
    save_loc = get(handles.edit_save_loc,'String');
    if save_loc(end) ~= '/' && save_loc(end) ~= '\'; save_loc =
[save_loc, '/']; end;
    handles.save_loc = save_loc;
end;
if ~isfield(handles,'compass_loc'); handles.compass_loc =
[handles.save_loc, 'compass/']; end;
compass_loc = handles.compass_loc;
if ~exist(compass_loc,'dir'); mkdir(compass_loc); end;

```

```

for k = 1:numel(tmp_vec);
    Vx = tmp_vec{k,1}(:,5); Vy = tmp_vec{k,1}(:,6);
    Figure(10); compass(Vx+sqrt(-1)*Vy); axis equal tight;
    % write the plotted Figure
    F = getframe(10);
    imwrite(F.cdata,[compass_loc,fname,'_compass_',sprintf('%02d.',k),type]);
end;
% % histogram and save
bin_width = pi/6;
angle_bins = -pi+bin_width/2 : bin_width : pi;
if ~isfield(handles,'histogram_loc'); handles.histogram_loc =
[handles.save_loc,'histogram/']; end;
hist_loc = handles.histogram_loc;
if ~exist(hist_loc,'dir'); mkdir(hist_loc); end;
for k = 1:numel(tmp_vec);
    angles = tmp_vec{k,1}(:,7);
    Figure(10); hist(angles,angle_bins);
    % write the plotted Figure
    F = getframe(10);
    imwrite(F.cdata,[hist_loc,fname,'_histogram_',sprintf('%02d.',k),type]);
end;
guidata(hObject,handles);

% % save those useful data before exiting
% --- Executes on button press in pushbutton_save_records.
function pushbutton_save_records_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton_save_records (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% % save these data
vpts_rect = handles.vpts_rect;
roi_mask = handles.roi_mask;
img_type = handles.img_type;
current_file_index = handles.current_file_index;
% % save these data at the location handles.mtrx_loc
if ~isfield(handles,'mtrx_loc'); handles.mtrx_loc = './roi_mask.mat'; end;
save(handles.mtrx_loc,'vpts_rect','roi_mask','current_file_index','img_type')
;
guidata(hObject,handles);

% % load the last record for further processing
% --- Executes on button press in pushbutton_load_last_record.
function pushbutton_load_last_record_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton_load_last_record (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% % location for the record
if ~isfield(handles,'save_loc'); save_loc =
get(handles.edit_save_loc,'String');
if save_loc(end) ~= '/' && save_loc(end) ~= '\'; save_loc =
[save_loc, '/']; end;

```

```
handles.save_loc = save_loc;
end;
if ~isfield(handles,'mtrx_loc'); handles.mtrx_loc = [handles.save_loc
'roi_mask.mat']; end;
%% load the record
load(handles.mtrx_loc);
%% init the handles
handles.vpts_rect = vpts_rect;
handles.roi_mask = roi_mask;
handles.img_type = img_type;
handles.current_file_index = current_file_index;

guidata(hObject,handles);

%% create roi_mask for image files in the dir
function roi_mask = roi_mask_create_struct(roi_mask,dir_name,type)
%% create the struct for all the files in the dir_name
%% each file has the struct:
%% .loc           %% location of the image file
%% .pctloc        %% location of the pct file storing the flow vectors
%% .roiflowvectors %% storing vectors within the rectangles
%% .rect          %% ROI rectangles on the images

L = numel(type);
%% get all the files within the dir
files = dir(dir_name);
for k = 1:numel(files);
    %% k-th files
    filesk = files(k);
    fnamek = filesk.name;
    %% if this file is a sub-dir
    if filesk.isdir;
        %% unuseful sub-dir maybe '.' or '..'
        if ~strcmp(fnamek, '.') && ~strcmp(fnamek, '..');
            %% create part of roi_mask for image files in the sub-dir
            dirk = [dir_name,fnamek, '/'];
            roi_mask = roi_mask_create_struct(roi_mask,dirk,type);
        end;
    %% if this is an image file
    elseif strcmp(fnamek(end-L+1:end),type)
        %% location of the image file
        tmp.loc = [dir_name,fnamek];
        %% location of the '.pct' file
        %% dir_name may be '.../face/';
        %% pct files are under '.../pct/', with the same name
        %% make sure the locations are arranged like this
        tmp.pctloc = [dir_name(1:end-5),'pct/',fnamek(1:end-L),'pct'];
        %% cell of flow vectors within ROI rects
        tmp.roiflowvectors = cell(0,0);
        %% rect matrix of size ?*4
        tmp.rect = zeros(0,4);
        %% update roi_mask
        %% this kind of updating may be slow with a lot of image files
    end;
end;
```

```

%         roi_mask = [roi_mask; tmp];
    end;
end;
% added
files = dir([dir_name, '*.', type]);
if ~isempty(files);
    fnames = {files.name};
    for k = [1 numel(fnames)];
        fnamek = fnames{k};
        tmp.name = fnamek(1:end-L+1);
        tmp.loc = [dir_name, fnamek];
        tmp.pctloc = [dir_name(1:end-5), 'pct/', fnamek(1:end-L), 'pct'];
        tmp.roiflowvectors = cell(0,0);
        tmp.rect = zeros(0,4);
        roi_mask = [roi_mask; tmp];
    end;
end;

% % load flow vectors from the pct file
function data = roi_mask_load_pct_data(loc)
% % these vectors start from the 26-th lines
% % ref. to the struct of the pct files
headerlines = 12+1 + 10+1 + 1;
% % open file
fid = fopen(loc, 'r');
% % read number flow of vectors
N = textscan(fid, '%d', 1, 'headerlines', headerlines-1);
N = N{1}; % % number of flow vectors
% % close file
fclose(fid);

fid = fopen(loc, 'r');
% % read N flow vectors
data = textscan(fid, '%f %f %f %f', N, 'headerlines', headerlines);
fclose(fid);
% % shape into N*4 matrix
data = [data{1} data{2} data{3} data{4}];

% % find the flow vectors within the rects
function roi_flow_vectors = roi_mask_roi_flow_vectors(pctdata, vpts_rect)
% threshold
thr = 0.05;
% % init
roi_flow_vectors = cell(size(vpts_rect,1),1);
% % calc vectors within the k-th rect
for k = 1:size(vpts_rect,1);
    % % bounds of coordinates
    xmin = vpts_rect(k,1);
    ymin = vpts_rect(k,2);
    xmax = vpts_rect(k,3) + xmin;
    ymax = vpts_rect(k,4) + ymin;
    % % flow vectors
    tmp = pctdata;

```

```
% % vectors within these bounds
tmp = tmp(tmp(:,1)>=xmin,:);
tmp = tmp(tmp(:,2)>=ymin,:);
tmp = tmp(tmp(:,3)<=xmax,:);
tmp = tmp(tmp(:,4)<=ymax,:);
% % flow vectors
vectors = tmp(:,3:4)-tmp(:,1:2);
idx = abs(vectors(:,1)) > thr | abs(vectors(:,2)) > thr;
vectors = vectors(idx,:);

tmp = tmp(idx,:);
% % calc angles
angles = atan2( vectors(:,2),vectors(:,1) );
% % vectors within the k-th rect
roi_flow_vectors{k,1} = [tmp vectors angles];
end;

% % show the vectors in a new Figure
% --- Executes on button press in pushbutton_show_flow.
function pushbutton_show_flow_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton_show_flow (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% % current image file
dex = handles.current_file_index;
roi_cell = handles.roi_mask{dex};
img = imread(roi_cell.loc);
% % shape the rects as a matrix
rects = [];
for i = 1:numel(roi_cell.roiflowvectors);
    rects = [rects; roi_cell.roiflowvectors{i}];
end;
% % calc [x,y,u,v] for quiver
x = rects(:,1);
y = rects(:,2);
u = rects(:,5);%u = rects(:,3) - rects(:,1);
v = rects(:,6);%v = rects(:,4) - rects(:,2);
% % narrow the range of the image
% % comment this part if you want to show the original image
xmin = max(floor(min(x))-30, 1);
ymin = max(floor(min(y))-30, 1);
xmax = min(ceil(max(x))+30, size(img,2));
ymax = min(ceil(max(y))+30, size(img,1));
img = img(ymin:ymax, xmin:xmax, :);
x = x - (xmin-1);
y = y - (ymin-1);
% % discard small vectors
% % comment this part if you want to show all the vectors
% thr = .05;
% index = (abs(u) > thr) | (abs(v) > thr);
% x = x(index); y = y(index);
% u = u(index); v = v(index);
```

```
% % show the ROI vectors
Figure(10); imshow(img); hold on;
quiver(x,y,u,v); hold off;
```

Create Dataset MATLAB code

```
function features = extract_mean_features(filename,rows)
% e.g.
% filename = 'D:\work\roi_mask.mat'
% rows = 1:8 or like [13 14]
rows = sort(rows(:)); % ascending row indices
if exist(filename,'file'); load(filename);
else features = [];
    error([' File ',filename,' does not exist ']);
end;

num_of_rows = numel(rows);
cols = [5 6 7]; % cols of [vx vy angle]
num_of_cols = numel(cols);

features = roi_mask;
for k = 1:numel(features);
    % ?*1 labels
    features{k}.labels = zeros(num_of_rows,1);
    % ?*3 features: vx vy angles (mean)
    if isempty(features{k}.roiflowvectors); % vectors not exist
        features{k}.features = [];
    else
        features{k}.features = zeros(num_of_rows,num_of_cols);
        vec = features{k}.roiflowvectors(rows);
        for i = 1:num_of_rows
            features{k}.features(i,:) = mean(vec{i}(:,cols),1);
        end;
    end;
    % remove some useless fields
    features{k} = rmfield(features{k},{'roiflowvectors','rect'});
end;
```

Appendices F

An Example of Defining and Learning the Facial Expression Model

```
1  #include <string >
2  #include "smile.h"
3  #include "smilearn.h"
4
5  using namespace std;
6
7  class FaceExp{
8  public:
9      FaceExp(void){};
10     ~FaceExp(void){};
11     void CopyCPT(int from, int to, int order);
12     void CopyDBNParameters(void);
13     void CreateDBNStructure(void);
14     void CreateTransitionNetStructure(void);
15     void LearnTransitionNet(void);
16     void LoadDataFile(string datafile);
17     void Run(string datafile);
18     void SaveDBN(string dynNetStr);
19
20     DSL_network dynNet;
21     DSL_network tNet;
22     DSL_dataset dataset;
23 };
```

```
1  #include "FaceExp.h"
2  #include <iostream >
3
4  //-----
5
6  void FaceExp::CopyCPT(int from, int to, int order){
7      DSL_Dmatrix probs;
8      probs = *(tNet.GetNode(from)->Definition()->GetMatrix());
9      if (order == 0){
10         dynNet.GetNode(to)->Definition()->SetDefinition(probs);
11     } else {
12         dynNet.GetNode(to)->TemporalDefinition()->SetTemporalMatrix(probs ,
13 order);
14     }
15 }
```

```

16
17 //-----
18
19 void FaceExp::CopyDBNParameters(void){
20 // Get the nodes from the dbn.
21 int BrowL_up_t = dynNet.FindNode("BrowL_up");
22 int BrowL_down_t = dynNet.FindNode("BrowL_down");
23 int BrowA_up_t = dynNet.FindNode("BrowA_up");
24 int BrowA_down_t = dynNet.FindNode("BrowA_down");
25 int NFL_up_t = dynNet.FindNode("NFL_up");
26 int NFL_down_t = dynNet.FindNode("NFL_down");
27 int NFA_up_t = dynNet.FindNode("NFA_up");
28 int NFA_down_t = dynNet.FindNode("NFA_down");
29 int LidsL_up_t = dynNet.FindNode("LidsL_up");
30 int LidsL_down_t = dynNet.FindNode("LidsL_down");
31 int LidsA_up_t = dynNet.FindNode("LidsA_up");
32 int LidsA_down_t = dynNet.FindNode("LidsA_down");
33 int ChinL_up_t = dynNet.FindNode("ChinL_up");
34 int ChinL_down_t = dynNet.FindNode("ChinL_down");
35 int ChinA_up_t = dynNet.FindNode("ChinA_up");
36 int ChinA_down_t = dynNet.FindNode("ChinA_down");
37 int MouthL_up_t = dynNet.FindNode("MouthL_up");
38 int MouthL_down_t = dynNet.FindNode("MouthL_down");
39 int MouthA_up_t = dynNet.FindNode("MouthA_up");
40 int MouthA_down_t = dynNet.FindNode("MouthA_down");
41 int UplipL_up_t = dynNet.FindNode("UplipL_up");
42 int UplipL_down_t = dynNet.FindNode("UplipL_down");
43 int UplipA_up_t = dynNet.FindNode("UplipA_up");
44 int UplipA_down_t = dynNet.FindNode("UplipA_down");
45 int LowlipL_up_t = dynNet.FindNode("LowlipL_up");
46 int LowlipL_down_t = dynNet.FindNode("LowlipL_down");
47 int LowlipA_up_t = dynNet.FindNode("LowlipA_up");
48 int LowlipA_down_t = dynNet.FindNode("LowlipA_down");
49 int lipL_up_t = dynNet.FindNode("lipL_up");
50 int lipL_down_t = dynNet.FindNode("lipL_down");
51 int lipA_up_t = dynNet.FindNode("lipA_up");
52 int lipA_down_t = dynNet.FindNode("lipA_down");
53 int lipcornersL_up_t = dynNet.FindNode("lipcornersL_up");
54 int lipcornersL_down_t = dynNet.FindNode("lipcornersL_down");
55 int lipcornersA_up_t = dynNet.FindNode("lipcornersA_up");
56 int lipcornersA_down_t = dynNet.FindNode("lipcornersA_down");
57
58 // Get the nodes from the trained network - Order 0.
59 int BrowL_up_t = dynNet.FindNode("BrowL_up");
60 int BrowL_down_t = dynNet.FindNode("BrowL_down");
61 int BrowA_up_t = dynNet.FindNode("BrowA_up");
62 int BrowA_down_t = dynNet.FindNode("BrowA_down");
63 int NFL_up_t = dynNet.FindNode("NFL_up");
64 int NFL_down_t = dynNet.FindNode("NFL_down");
65 int NFA_up_t = dynNet.FindNode("NFA_up");
66 int NFA_down_t = dynNet.FindNode("NFA_down");
67 int LidsL_up_t = dynNet.FindNode("LidsL_up");
68 int LidsL_down_t = dynNet.FindNode("LidsL_down");

```



```
69     int LidsA_up_t = dynNet.FindNode("LidsA_up");
70     int LidsA_down_t = dynNet.FindNode("LidsA_down");
71     int ChinL_up_t = dynNet.FindNode("ChinL_up");
72     int ChinL_down_t = dynNet.FindNode("ChinL_down");
73     int ChinA_up_t = dynNet.FindNode("ChinA_up");
74     int ChinA_down_t = dynNet.FindNode("ChinA_down");
75     int MouthL_up_t = dynNet.FindNode("MouthL_up");
76     int MouthL_down_t = dynNet.FindNode("MouthL_down");
77     int MouthA_up_t = dynNet.FindNode("MouthA_up");
78     int MouthA_down_t = dynNet.FindNode("MouthA_down");
79     int UplipL_up_t = dynNet.FindNode("UplipL_up");
80     int UplipL_down_t = dynNet.FindNode("UplipL_down");
81     int UplipA_up_t = dynNet.FindNode("UplipA_up");
82     int UplipA_down_t = dynNet.FindNode("UplipA_down");
83     int LowlipL_up_t = dynNet.FindNode("LowlipL_up");
84     int LowlipL_down_t = dynNet.FindNode("LowlipL_down");
85     int LowlipA_up_t = dynNet.FindNode("LowlipA_up");
86     int LowlipA_down_t = dynNet.FindNode("LowlipA_down");
87     int lipL_up_t = dynNet.FindNode("lipL_up");
88     int lipL_down_t = dynNet.FindNode("lipL_down");
89     int lipA_up_t = dynNet.FindNode("lipA_up");
90     int lipA_down_t = dynNet.FindNode("lipA_down");
91     int lipcornersL_up_t = dynNet.FindNode("lipcornersL_up");
92     int lipcornersL_down_t = dynNet.FindNode("lipcornersL_down");
93     int lipcornersA_up_t = dynNet.FindNode("lipcornersA_up");
94     int lipcornersA_down_t = dynNet.FindNode("lipcornersA_down");
95
96     // Copy the CPTs from the trained network to the dynamic network -
97     Order 0.
98
99     CopyCPT(BrowL_up_tNet_0 , BrowL_up_t , 0);
100    CopyCPT(BrowL_down_tNet_0 , BrowL_down_t , 0);
101    CopyCPT(BrowA_up_tNet_0 , BrowA_up_t , 0);
102    CopyCPT(BrowA_down_tNet_0 , BrowA_down_t , 0);
103    CopyCPT(NFL_up_tNet_0 , NFL_up_t , 0);
104    CopyCPT(NFL_down_tNet_0 , NFL_down_t , 0);
105    CopyCPT(NFA_up_tNet_0 , NFA_up_t , 0);
106    CopyCPT(NFA_down_tNet_0 , NFA_down_t , 0);
107    CopyCPT(LidsL_up_tNet_0 , LidsL_up_t , 0);
108    CopyCPT(LidsL_down_tNet_0 , LidsL_down_t , 0);
109    CopyCPT(LidsA_up_tNet_0 , LidsA_up_t , 0);
110    CopyCPT(LidsA_down_tNet_0 , LidsA_down_t , 0);
111    CopyCPT(ChinL_up_tNet_0 , ChinL_up_t , 0);
112    CopyCPT(ChinL_down_tNet_0 , ChinL_down_t , 0);
113    CopyCPT(ChinA_up_tNet_0 , ChinA_up_t , 0);
114    CopyCPT(ChinA_down_tNet_0 , ChinA_down_t , 0);
115    CopyCPT(MouthL_up_tNet_0 , MouthL_up_t , 0);
116    CopyCPT(MouthL_down_tNet_0 , MouthL_down_t , 0);
117    CopyCPT(MouthA_up_tNet_0 , MouthA_up_t , 0);
118    CopyCPT(MouthA_down_tNet_0 , MouthA_down_t , 0);
119    CopyCPT(UplipL_up_tNet_0 , UplipL_up_t , 0);
120    CopyCPT(UplipL_down_tNet_0 , UplipL_down_t , 0);
121    CopyCPT(UplipA_up_tNet_0 , UplipA_up_t , 0);
122    CopyCPT(UplipA_down_tNet_0 , UplipA_down_t , 0);
```

```

123 CopyCPT(LowlipL_up_tNet_0 , LowlipL_up_t , 0);
124 CopyCPT(LowlipL_down_tNet_0 , LowlipL_down_t , 0);
125 CopyCPT(LowlipA_up_tNet_0 , LowlipA_up_t , 0);
126 CopyCPT(LowlipA_down_tNet_0 , LowlipA_down_t , 0);
127 CopyCPT(lipL_up_tNet_0 , lipL_up_t , 0);
128 CopyCPT(lipL_down_tNet_0 , lipL_down_t , 0);
129 CopyCPT(lipA_up_tNet_0 , lipA_up_t , 0);
130 CopyCPT(lipA_down_tNet_0 , lipA_down_t , 0);
131 CopyCPT(lipcornersL_up_tNet_0 , lipcornersL_up_t , 0);
132 CopyCPT(lipcornersL_down_tNet_0 , lipcornersL_down_t , 0);
133 CopyCPT(lipcornersA_up_tNet_0 , lipcornersA_up_t , 0);
134 CopyCPT(lipcornersA_down_tNet_0 , lipcornersA_down_t , 0);
135
136 // Order 1.
137 CopyCPT(BrowL_up_tNet_1 , BrowL_up_t , 1);
138 CopyCPT(BrowL_down_tNet_1 , BrowL_down_t , 1);
139 CopyCPT(BrowA_up_tNet_1 , BrowA_up_t , 1);
140 CopyCPT(BrowA_down_tNet_1 , BrowA_down_t , 1);
141 CopyCPT(NFL_up_tNet_1 , NFL_up_t , 1);
142 CopyCPT(NFL_down_tNet_1 , NFL_down_t , 1);
143 CopyCPT(NFA_up_tNet_1 , NFA_up_t , 1);
144 CopyCPT(NFA_down_tNet_1 , NFA_down_t , 1);
145 CopyCPT(LidsL_up_tNet_1 , LidsL_up_t , 1);
146 CopyCPT(LidsL_down_tNet_1 , LidsL_down_t , 1);
147 CopyCPT(LidsA_up_tNet_1 , LidsA_up_t , 1);
148 CopyCPT(LidsA_down_tNet_1 , LidsA_down_t , 1);
149 CopyCPT(ChinL_up_tNet_1 , ChinL_up_t , 1);
150 CopyCPT(ChinL_down_tNet_1 , ChinL_down_t , 1);
151 CopyCPT(ChinA_up_tNet_1 , ChinA_up_t , 1);
152 CopyCPT(ChinA_down_tNet_1 , ChinA_down_t , 1);
153 CopyCPT(MouthL_up_tNet_1 , MouthL_up_t , 1);
154 CopyCPT(MouthL_down_tNet_1 , MouthL_down_t , 1);
155 CopyCPT(MouthA_up_tNet_1 , MouthA_up_t , 1);
156 CopyCPT(MouthA_down_tNet_1 , MouthA_down_t , 1);
157 CopyCPT(UplipL_up_tNet_1 , UplipL_up_t , 1);
158 CopyCPT(UplipL_down_tNet_1 , UplipL_down_t , 1);
159 CopyCPT(UplipA_up_tNet_1 , UplipA_up_t , 1);
160 CopyCPT(UplipA_down_tNet_1 , UplipA_down_t , 1);
161 CopyCPT(LowlipL_up_tNet_1 , LowlipL_up_t , 1);
162 CopyCPT(LowlipL_down_tNet_1 , LowlipL_down_t , 1);
163 CopyCPT(LowlipA_up_tNet_1 , LowlipA_up_t , 1);
164 CopyCPT(LowlipA_down_tNet_1 , LowlipA_down_t , 1);
165 CopyCPT(lipL_up_tNet_1 , lipL_up_t , 1);
166 CopyCPT(lipL_down_tNet_1 , lipL_down_t , 1);
167 CopyCPT(lipA_up_tNet_1 , lipA_up_t , 1);
168 CopyCPT(lipA_down_tNet_1 , lipA_down_t , 1);
169 CopyCPT(lipcornersL_up_tNet_1 , lipcornersL_up_t , 1);
170 CopyCPT(lipcornersL_down_tNet_1 , lipcornersL_down_t , 1);
171 CopyCPT(lipcornersA_up_tNet_1 , lipcornersA_up_t , 1);
172 CopyCPT(lipcornersA_down_tNet_1 , lipcornersA_down_t , 1);
173 };
174
175 //-----

```

```

176
177 void FaceExp::CreateDBNStructure(void){
178
179     // Initialize nodes for the dynamic network.
180     int BrowL_up_t = dynNet.AddNode(DSL_CPT , "BrowL_up");
181     int BrowL_down_t = dynNet.AddNode(DSL_CPT , "BrowL_down");
182     int BrowA_up_t = dynNet.AddNode(DSL_CPT , "BrowA_up");
183     int BrowA_down_t = dynNet.AddNode(DSL_CPT , "BrowA_down");
184     int NFL_up_t = dynNet.AddNode(DSL_CPT , "NFL_up");
185     int NFL_down_t = dynNet.AddNode(DSL_CPT , "NFL_down");
186     int NFA_up_t = dynNet.AddNode(DSL_CPT , "NFA_up");
187     int NFA_down_t = dynNet.AddNode(DSL_CPT , "NFA_down");
188     int LidsL_up_t = dynNet.AddNode(DSL_CPT , "LidsL_up");
189     int LidsL_down_t = dynNet.AddNode(DSL_CPT , "LidsL_down");
190     int LidsA_up_t = dynNet.AddNode(DSL_CPT , "LidsA_up");
191     int LidsA_down_t = dynNet.AddNode(DSL_CPT , "LidsA_down");
192     int ChinL_up_t = dynNet.AddNode(DSL_CPT , "ChinL_up");
193     int ChinL_down_t = dynNet.AddNode(DSL_CPT , "ChinL_down");
194     int ChinA_up_t = dynNet.AddNode(DSL_CPT , "ChinA_up");
195     int ChinA_down_t = dynNet.AddNode(DSL_CPT , "ChinA_down");
196     int MouthL_up_t = dynNet.AddNode(DSL_CPT , "MouthL_up");
197     int MouthL_down_t = dynNet.AddNode(DSL_CPT , "MouthL_down");
198     int MouthA_up_t = dynNet.AddNode(DSL_CPT , "MouthA_up");
199     int MouthA_down_t = dynNet.AddNode(DSL_CPT , "MouthA_down");
200     int UplipL_up_t = dynNet.AddNode(DSL_CPT , "UplipL_up");
201     int UplipL_down_t = dynNet.AddNode(DSL_CPT , "UplipL_down");
202     int UplipA_up_t = dynNet.AddNode(DSL_CPT , "UplipA_up");
203     int UplipA_down_t = dynNet.AddNode(DSL_CPT , "UplipA_down");
204     int LowlipL_up_t = dynNet.AddNode(DSL_CPT , "LowlipL_up");
205     int LowlipL_down_t = dynNet.AddNode(DSL_CPT , "LowlipL_down");
206     int LowlipA_up_t = dynNet.AddNode(DSL_CPT , "LowlipA_up");
207     int LowlipA_down_t = dynNet.AddNode(DSL_CPT , "LowlipA_down");
208     int lipL_up_t = dynNet.AddNode(DSL_CPT , "lipL_up");
209     int lipL_down_t = dynNet.AddNode(DSL_CPT , "lipL_down");
210     int lipA_up_t = dynNet.AddNode(DSL_CPT , "lipA_up");
211     int lipA_down_t = dynNet.AddNode(DSL_CPT , "lipA_down");
212     int lipcornersL_up_t = dynNet.AddNode(DSL_CPT , "lipcornersL_up");
213     int lipcornersL_down_t = dynNet.AddNode(DSL_CPT , "lipcornersL_down");
214     int lipcornersA_up_t = dynNet.AddNode(DSL_CPT , "lipcornersA_up");
215     int lipcornersA_down_t = dynNet.AddNode(DSL_CPT , "lipcornersA_down");
216
217
218     // Set the outcomenames.
219     int handle = dynNet.GetFirstNode();
220     while (handle != DSL_OUT_OF_RANGE){
221         // Map variable in dataset to dynNet.
222         string id(dynNet.GetNode(handle)->Info().Header().GetId());
223         int v = dataset.FindVariable(id);
224
225         // Get the stateNames.
226         vector <string > stateNamesStr = dataset.GetStateNames(v);
227         DSL_stringArray stateNames;
228         vector <string >::iterator iter = stateNamesStr.begin();
229         for(iter; iter < stateNamesStr.end(); iter++){

```

```

230         stateNames.Add(const_cast <char*>(iter->c_str()));
231     }
232     dynNet.GetNode(handle)->Definition()-
233 >SetNumberOfOutcomes(stateNames);
234
235     // Go to next node in dynNet.
236     handle = dynNet.GetNextNode(handle);
237 }
238
239 // Set temporal types.
240 dynNet.SetTemporalType(BrowL_up_t , dsl_plateNode);
241 dynNet.SetTemporalType(BrowL_down_t , dsl_plateNode);
242 dynNet.SetTemporalType(BrowA_up_t , dsl_plateNode);
243 dynNet.SetTemporalType(BrowA_down_t , dsl_plateNode);
244 dynNet.SetTemporalType(NFL_up_t , dsl_plateNode);
245 dynNet.SetTemporalType(NFL_down_t , dsl_plateNode);
246 dynNet.SetTemporalType(NFA_up_t , dsl_plateNode);
247 dynNet.SetTemporalType(NFA_down_t , dsl_plateNode);
248 dynNet.SetTemporalType(LidsL_up_t , dsl_plateNode);
249 dynNet.SetTemporalType(LidsL_down_t , dsl_plateNode);
250 dynNet.SetTemporalType(LidsA_up_t , dsl_plateNode);
251 dynNet.SetTemporalType(LidsA_down_t , dsl_plateNode);
252 dynNet.SetTemporalType(ChinL_up_t , dsl_plateNode);
253 dynNet.SetTemporalType(ChinL_down_t , dsl_plateNode);
254 dynNet.SetTemporalType(ChinA_up_t , dsl_plateNode);
255 dynNet.SetTemporalType(ChinA_down_t , dsl_plateNode);
256 dynNet.SetTemporalType(MouthL_up_t , dsl_plateNode);
257 dynNet.SetTemporalType(MouthL_down_t , dsl_plateNode);
258 dynNet.SetTemporalType(MouthA_up_t , dsl_plateNode);
259 dynNet.SetTemporalType(MoythA_down_t , dsl_plateNode);
260 dynNet.SetTemporalType(UplipL_up_t , dsl_plateNode);
261 dynNet.SetTemporalType(UplipL_down_t , dsl_plateNode);
262 dynNet.SetTemporalType(UplipA_up_t , dsl_plateNode);
263 dynNet.SetTemporalType(UplipA_down_t , dsl_plateNode);
264 dynNet.SetTemporalType(LowlipL_up_t , dsl_plateNode);
265 dynNet.SetTemporalType(LpwlipL_down_t , dsl_plateNode);
266 dynNet.SetTemporalType(LowlipA_up_t , dsl_plateNode);
267 dynNet.SetTemporalType(LowlipA_down_t , dsl_plateNode);
268 dynNet.SetTemporalType(lipL_up_t , dsl_plateNode);
269 dynNet.SetTemporalType(lipL_down_t , dsl_plateNode);
270 dynNet.SetTemporalType(lipA_up_t , dsl_plateNode);
271 dynNet.SetTemporalType(lipA_down_t , dsl_plateNode);
272 dynNet.SetTemporalType(lipcornersL_up_t , dsl_plateNode);
273 dynNet.SetTemporalType(lipcornersL_down_t , dsl_plateNode);
274 dynNet.SetTemporalType(lipcornersA_up_t , dsl_plateNode);
275 dynNet.SetTemporalType(lipcornersA_down_t , dsl_plateNode);
276
277 // Add arcs - Order 0.
278
279 dynNet.AssArc(BrowL_up_t , dsl_plate_AU1,AU2,AU4);
280 dynNet.AssArc(BrowL_down_t , dsl_plate_AU1,AU2,AU4);
281 dynNet.AssArc(BrowA_up_t , dsl_plate_AU1,AU2,AU4);
282 dynNet.AssArc(BrowA_down_t , dsl_plateAU1,AU2,AU4);

```

```
283 dynNet.AssArc(NFL_up_t , dsl_plate_AU6,AU9);
284 dynNet.AssArc(NFL_down_t , dsl_plat_AU6,AU9);
285 dynNet.AssArc(NFA_up_t , dsl_plate_AU6,AU9);
286 dynNet.AssArc(NFA_down_t , dsl_plate_AU6,AU9);
287 dynNet.AssArc(LidsL_up_t , dsl_plate_AU5,AU7);
288 dynNet.AssArc(LidsL_down_t , dsl_plate_AU5,AU7);
289 dynNet.AssArc(LidsA_up_t , dsl_plate_AU5,AU7);
290 dynNet.AssArc(LidsA_down_t , dsl_plate_AU5,AU7);
291 dynNet.AssArc(ChinL_up_t , dsl_plate_AU17);
292 dynNet.AssArc(ChinL_down_t , dsl_plate_AU17);
293 dynNet.AssArc(ChinA_up_t , dsl_plate_AU17);
294 dynNet.AssArc(ChinA_down_t , dsl_plate_AU17);
295 dynNet.AssArc(MouthL_up_t , dsl_plate_AU25,AU26,AU27);
296 dynNet.AssArc(MouthL_down_t , dsl_plate_AU25,AU26,AU27);
297 dynNet.AssArc(MouthA_up_t , dsl_plate_AU25,AU26,AU27);
298 dynNet.AssArc(MoythA_down_t , dsl_plate_AU25,AU26,AU27);
299 dynNet.AssArc(UplipL_up_t , dsl_plate_AU10);
300 dynNet.AssArc(UplipL_down_t , dsl_plate_AU10);
301 dynNet.AssArc(UplipA_up_t , dsl_plate_AU10);
302 dynNet.AssArc(UplipA_down_t , dsl_plate_AU10);
303 dynNet.AssArc(LowlipL_up_t , dsl_plate_AU16);
304 dynNet.AssArc(LpwlipL_down_t , dsl_plate_AU16);
305 dynNet.AssArc(LowlipA_up_t , dsl_plate_AU16);
306 dynNet.AssArc(LowlipA_down_t , dsl_plate_AU16);
307 dynNet.AssArc(lipL_up_t , dsl_plate_AU20,AU23,AU24);
308 dynNet.AssArc(lipL_down_t , dsl_plate_AU20,AU23,AU24);
309 dynNet.AssArc(lipA_up_t , dsl_plate_AU20,AU23,AU24);
310 dynNet.AssArc(lipA_down_t , dsl_plate_AU20,AU23,AU24);
311 dynNet.AssArc(lipcornersL_up_t , dsl_plate_AU12,AU15);
312 dynNet.AssArc(lipcornersL_down_t , dsl_plate_AU12,AU15);
313 dynNet.AssArc(lipcornersA_up_t , dsl_plate_AU12,AU15);
314 dynNet.AssArc(lipcornersA_down_t , dsl_plate_AU12,AU15);
315
316 // Order 1.
317 dynNet.AssArc(BrowL_up_t , dsl_plate_AU1,AU2,AU4, 1);
318 dynNet.AssArc(BrowL_down_t , dsl_plate_AU1,AU2,AU4, 1);
319 dynNet.AssArc(BrowA_up_t , dsl_plate_AU1,AU2,AU4, 1);
320 dynNet.AssArc(BrowA_down_t , dsl_plateAU1,AU2,AU4, 1);
321 dynNet.AssArc(NFL_up_t , dsl_plate_AU6,AU9, 1);
322 dynNet.AssArc(NFL_down_t , dsl_plat_AU6,AU9, 1);
323 dynNet.AssArc(NFA_up_t , dsl_plate_AU6,AU9, 1);
324 dynNet.AssArc(NFA_down_t , dsl_plate_AU6,AU9, 1);
325 dynNet.AssArc(LidsL_up_t , dsl_plate_AU5,AU7, 1);
326 dynNet.AssArc(LidsL_down_t , dsl_plate_AU5,AU7, 1);
327 dynNet.AssArc(LidsA_up_t , dsl_plate_AU5,AU7, 1);
328 dynNet.AssArc(LidsA_down_t , dsl_plate_AU5,AU7, 1);
329 dynNet.AssArc(ChinL_up_t , dsl_plate_AU17, 1);
330 dynNet.AssArc(ChinL_down_t , dsl_plate_AU17, 1);
331 dynNet.AssArc(ChinA_up_t , dsl_plate_AU17, 1);
332 dynNet.AssArc(ChinA_down_t , dsl_plate_AU17, 1);
333 dynNet.AssArc(MouthL_up_t , dsl_plate_AU25,AU26,AU27, 1);
334 dynNet.AssArc(MouthL_down_t , dsl_plate_AU25,AU26,AU27, 1);
335 dynNet.AssArc(MouthA_up_t , dsl_plate_AU25,AU26,AU27, 1);
336 dynNet.AssArc(MoythA_down_t , dsl_plate_AU25,AU26,AU27, 1);
```

```

337     dynNet.AssArc(UplipL_up_t , dsl_plate_AU10, 1);
338     dynNet.AssArc(UplipL_down_t , dsl_plate_AU10, 1);
339     dynNet.AssArc(UplipA_up_t , dsl_plate_AU10, 1);
340     dynNet.AssArc(UplipA_down_t , dsl_plate_AU10, 1);
341     dynNet.AssArc(LowlipL_up_t , dsl_plate_AU16, 1);
342     dynNet.AssArc(LowlipL_down_t , dsl_plate_AU16, 1);
343     dynNet.AssArc(LowlipA_up_t , dsl_plate_AU16, 1);
344     dynNet.AssArc(LowlipA_down_t , dsl_plate_AU16, 1);
345     dynNet.AssArc(lipL_up_t , dsl_plate_AU20,AU23,AU24, 1);
346     dynNet.AssArc(lipL_down_t , dsl_plate_AU20,AU23,AU24, 1);
347     dynNet.AssArc(lipA_up_t , dsl_plate_AU20,AU23,AU24, 1);
348     dynNet.AssArc(lipA_down_t , dsl_plate_AU20,AU23,AU24, 1);
349     dynNet.AssArc(lipcornersL_up_t , dsl_plate_AU12,AU15, 1);
350     dynNet.AssArc(lipcornersL_down_t , dsl_plate_AU12,AU15, 1);
351     dynNet.AssArc(lipcornersA_up_t , dsl_plate_AU12,AU15, 1);
352     dynNet.AssArc(lipcornersA_down_t , dsl_plate_AU12,AU15,1);
353
354     // Set number of time-slices.
355     dynNet.SetNumberOfSlices(2);
356 }
357
358 //-----
359
360 void FaceExp::CreateTransitionNetStructure(void){
361
362     // Initialize.
363     int prevSlices = dynNet.GetNumberOfSlices();
364     dynNet.SetNumberOfSlices(2);
365     dynNet.UnrollNetwork(tNet);
366     dynNet.SetNumberOfSlices(prevSlices);
367
368     // Set [DSL_NOLEARN = ALL] userproperty.
369     DSL_node* temp = tNet.GetNode(tNet.FindNode("Shock_1"))
370     temp->Info().UserProperties().AddProperty("DSL_NOLEARN", "ALL");
371 }
372
373 //-----
374
375 void FaceExp::LearnTransitionNet(void){
376     // Map network to dataset.
377     for (int v = 0; v < dataset.NumVariables(); v++){
378         dataset.SetHandle(v, tNet.FindNode(dataset.GetId(v).c_str()));
379     }
380
381     // Learn the TransitionNet.
382     DSL_em learnParams;
383     learnParams.SetRandomizeParameters(false);
384     learnParams.Learn(dataset , tNet);
385 }
386
387 //-----
388
389 void FaceExp::LoadDataFile(string datafile){

```

```
390
391 // Initialize.
392 DSL_textParser parser;
393
394 // Parse the data file.
395 parser.SetUseHeader(true);
396 parser.SetTypesSpecified(false);
397 int result = parser.Parse(datafile.c_str());
398 if (result != DSL_OKAY) return result;
399
400 // Obtain the data.
414 dataset = parser.GetDataset();
402 }
403
404 //-----
405
406 void FaceExp::Run(string datafile){
407     cout << "Load-ata/ile..." << endl;
408     LoadDataFile(datafile);
409
410     cout << "Create-BN tructure..." << endl;
411     CreateDBNStructure();
412
413     cout << "Create Net tructure..." << endl;
414     CreateTransitionNetStructure();
415
416     cout << "Learn Net tructure..." << endl;
417     LearnTransitionNet();
418
419     cout << "Copy arameters o-BN..." << endl;
420     CopyDBNParameters();
421     SaveDBN("cv_steady_dbn_learned.xdsl");
422     cout << "Done..." << endl;
423 }
424
425 //-----
426
427 void FaceExp::SaveDBN(string dynNetStr){
428     return dynNet.WriteFile(dynNetStr.c_str());
429 }
430
431 //-----
```

Appendices G

A Bayesian Approach to Recognise Facial Expressions using Vector Flows

A Bayesian Approach to Recognise Facial Expressions using Vector Flows

Xiaofan Sun, Leon Rothkrantz and Pascal Wiggers

Man-Machine-Interaction Group,
Faculty of Electrical Engineering, Mathematics and Computer science,
Delft University of Technology
Mekelweg 4 2628CD Delft, The Netherlands

E-mail: momomisswy@hotmail.com, { L.J.M.Rothkrantz, P.Wiggers}@tudelft.nl

Abstract: *Facial expressions play an important role in human nonverbal communication. They can be generated by activation and dilatation of facial muscles. In this paper we describe a system to recognize facial expressions automatically. Special Length/Angles in the face have been selected to extract features from the vector flow of visual muscle activity. To classify facial expressions Bayesian Networks have been used. The classifier has been trained and tested on video recordings from the Cohn Kanade database. It contains recordings from the six basic emotions as defined by Ekman. The model and results of testing are reported in the paper.*

Key words: *Bayesian networks, Automatic Recognition of Facial Expressions, Vector Flow, Cohn Kanade database.*

INTRODUCTION

Facial expressions play an important role in human communication. Humans are able to convey their emotional state by facial expressions. Facial expressions are also used as paralinguistic cues to regulate our conversation. Facial expressions can be used to complete our verbal communication or to reduce the ambiguity in verbal communication. The research of facial expressions has a long history. Darwin pointed to the biological roots of facial expressions and the role they play in the survival of the species including human beings. Ekman [1] defined 6 basic emotional facial expression, happiness, sadness, disgust, anger, surprise and fear and claims that these expressions are universal. Most emotions are blended emotions and according to Ekman a combination of the basic emotions. Facial expressions are the result of activation/dilatation of one or more of the 43 facial muscles. Unfortunately from an observation point of view the activation of facial muscles is not observable directly. The activation of facial muscles can be observed by facial movements. Ekman defined 43 basic movements in the face called Action Units. He claims that every facial expression can be described in terms of Action Units. He defined a facial action coding scheme called FACS. That system has been used by

many researchers, especially those with a psychological background and focus of interest and many facial expressions are coded by the FACS system. One of the well known databases of video recordings of facial expressions is the Cohn Kanade database. The apex of the facial expressions in those recordings is fully described by the activation of the AUs.

The FACS system is based on human observations and the labeling process is manual. For many years, researchers designed systems to classify facial expressions in an automated way. Our system fits in that tradition. The innovative aspect of our system is that we first try to recognize the activation level of involved AUs. Next we can use the knowledge developed in the psychological domain to classify the emotions.

Many methods have been used in the past to classify emotional facial expressions, such as Artificial Neural Networks, Hidden Markov models, (dynamic) Bayesian networks, Support Vector machines etc. [2,3]. Most systems are focusing on the changing contours of the mouth, eyes and eyebrows. Special fiducially points have been defined on the contours, such as corners of the eyes and the mouth. Most methods try to localize and track those points. In case of happiness the corners of the mouth are going upwards and in case of sadness downwards. We use a different probabilistic approach. Facial expressions in video recordings can be visualized as vector flows. We define special Region Of Interest (ROI) around the underlying facial muscles. In those Length/Angles we extract parameters from the gradient field (average length and angle of direction). These parameters are fed into our Bayesian network to compute the probability of the six basic expressions. To train our system we used the data from the Cohn Kanade database [4].

The outline of this paper is as follows. In the next section we present an overview of related work. Next we define our models. Then we will present the results of our experiments and we will end with the conclusion.

RELATED WORK

In previous research the most used facial expression classification tool is FACS (Facial Action Coding System) by Ekman, Friesen and Hager [1], which is intended for describing all visually detectable changes on the face produced by facial muscle activity. The FACS system is used by human observers, after extensive training, to recognize and classify subtle facial actions [1]. Facial actions are described with objective and emotion-independent Action Units (AUs), depicted with abbreviations such as AU1, AU4 or AU1+4.

To recognize facial expression by AUs some groups use Gaussian Tree-Augmented Naive Bayes (TAN) to learn the dependencies among different facial motion features in order to classify facial expressions [3] However, due to TAN's structure limitations, it cannot handle complex relationships between facial features, as well as temporal changes. Zhang and Ji exploit a BN to classify six basic facial expressions with a dynamic fusion strategy.

Lien et al. [3] explored HMMs for facial AU recognition. Tian et al. [3] presented an NN-based approach, in which upper face AUs and lower face AUs are considered in NN construction. GU and Ji [6] use a similar idea for facial event classification such as fatigue. Cohen et al. [3] further

propose a classification driven stochastic structure search (SSS) algorithm for learning a BN classifier to recognize facial expression from both labeled and unlabeled data.

Most attempts on the representation of visual information for facial expression have focused on optical flow analysis from facial action [6], [8], [10], where optical flow is used to either model muscle activities or estimate the displacements of feature points. In this paper optical flow is used to estimate the motion of features in defined ROI. Lien et al. explored HMMs for facial AU recognition. Since each AU or AU combination associates with one HMM, the approach is infeasible for covering a great number of potential AU combinations involved in facial expressions. In this paper we first created a BN model to represent the causal relations between the ROI and facial AUs then extend the BN to a DBN model for modeling the dynamic behaviors of facial expressions in frame sequences. Colmenarez et al. [5] presented a Bayesian probabilistic approach to recognizing the face and facial expression but in this paper the works focus on recognizing facial AUs instead of facial expression.

Model

We mentioned in the introduction that facial expressions are generated by activation of facial muscles. The visual results of muscle activation are changing contours of the mouth, eye and eyebrow. But we also observe changing texture and position of wrinkles in the face. Many emotional facial expressions are characterized by the shape of the mouth, eye and eyebrow, including some wrinkles Length/Angles. In fact many facial muscles are in the length/angle around the mouth, eyes and eyebrow. The activity of facial muscles is limited to specific Length/Angles. To study facial movements we define special region of interest (ROIs). The idea is that one muscle activity and corresponding AU is limited to one ROI. But more than one muscle can be active in one ROI. In Fig 1 the ROIs are displayed.

The next step is to model the movement in facial expressions. We consider the movement of pixels in consecutive frames. This movement can be visualized by a vector field; every vector represents the speed and direction of movements. We used the Lucas Kanade algorithm [6] to compute the vector flow from the video recordings. As mentioned in the introduction we use the Cohn Kanade database. In that database about 70 persons show facial expressions from the 6 basic emotions starting from the neutral position up to the apex, showing the maximal intensity.

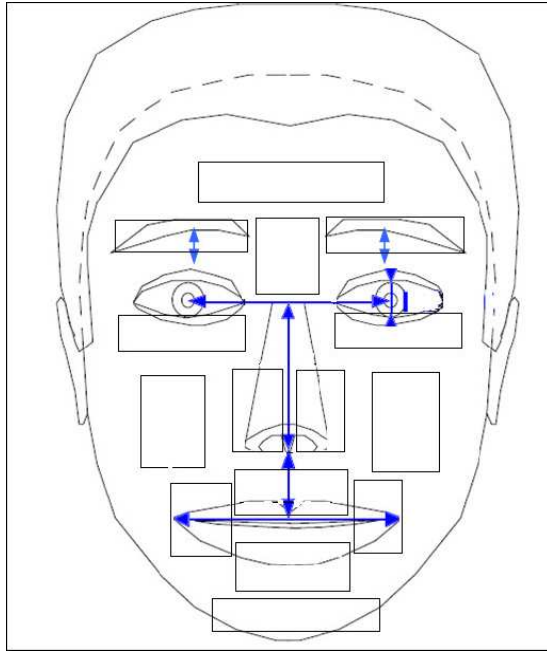


Figure 1. Model of the length/angle of Interest.

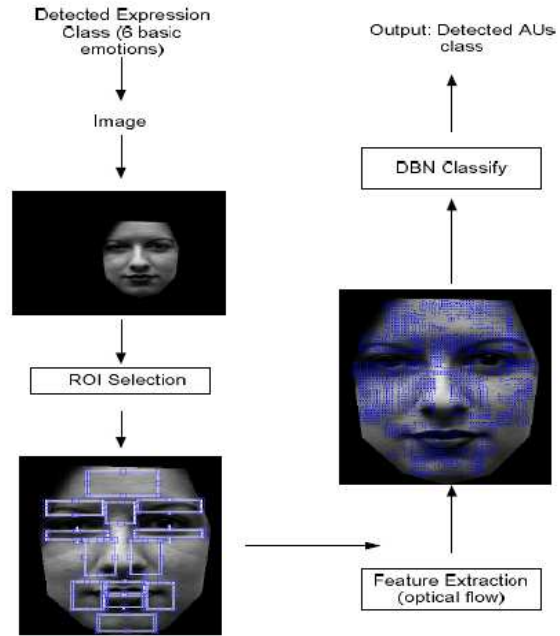


Figure 2. Flowchart of the data analysis.

We are only interested in facial movements. So the assumption is that the test persons don't move their heads. Unfortunately this assumption is violated many times, so we have to correct for head movements. We used the Active Appearance model [5] to localise the face and contours of eyes, eyebrow and mouth. We removed the background and hairs and selected only the mask of the face. This guarantees that our vector flow is concentrated of facial movements and not from the position of the head, hair etc. (see Figure 2).

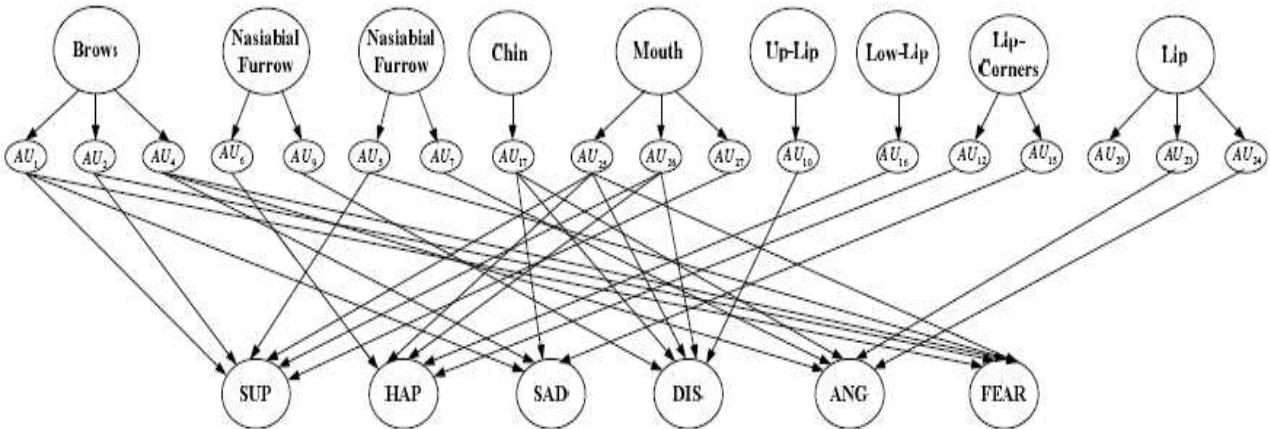


Figure 3. 3-layered Bayesian model.

Experimental Results

The selection of parameters as input of our Bayesian network is a complex process. The Lukas Kanade algorithm has many parameters to choose. Also the choice of ROI is arbitrary. To validate our choices we computed for every of the six basic emotions and ROI the average length and orientation of the vector flow. In Fig 4 we displayed the results. We observe a lot of variation and the average vectors have the expected size and orientation. For example in case of happiness the corners of the mouth are pulled upwards so the vector should point in North direction. In case of sadness we expect the opposite direction. In case of Fear and Surprise the eyes are fully open and in case of sadness only the upper eyelid is falling down. We observe the expected results in the in the displayed vector field.

A Bayesian network is a probabilistic belief network that is composed of variable nodes (chance, decision, deterministic) and directed arcs (a directed acyclic graph) or temporal arcs. Each node stands for a random variable with discrete values, and each arc has been used to present a dependence (causal) relationship between two nodes. To implement our Bayesian network model we used GeNIe developed by M. Druzdzel at Pittsburg University [7]. To compute the probabilities in the Conditional Probability tables we trained the network on the Cohn Kanade database. The GeNIe tools offers a learning algorithm based on the gradient descend method or Expectation maximisation (EM) method.

As shown in Table 1, the image sequences were assigned to training and testing sets. In the first set S1, the sequences were randomly selected, so the same subjects maybe appeared in both training and testing sets. But in the second set S2 no subject were allowed to appear in both training and testing sets, that is to say, the training data and testing data were totally different. The way in assigning training and testing sets is the same as Cohn's in [6].

Table 1. Splitting of dataset in training set and test.

Data Set		number of Sequences	Single AUs																		
			1	2	4	5	6	7	9	10	12	15	16	17	20	23	24	25	25	26	27
S1	Train	689	23	34	24	37	22	21	19	17	24	26	29	28	32	17	19	19	28	27	28
	Test	184	9	10	7	13	10	9	11	10	12	10	10	11	10	9	7	8	9	9	10
S2	Train	505	26	38	25	29	29	26	22	19	22	23	19	18	34	33	28	25	29	29	31
	Test	150	7	8	6	5	6	7	8	6	9	7	8	6	8	9	9	11	11	9	10

Finally, we apply the BN model to recognize single AUs and some important AU combinations. The average classification rate for the single AUs are between 80% and 90% and for the AU combinations are above 90%.

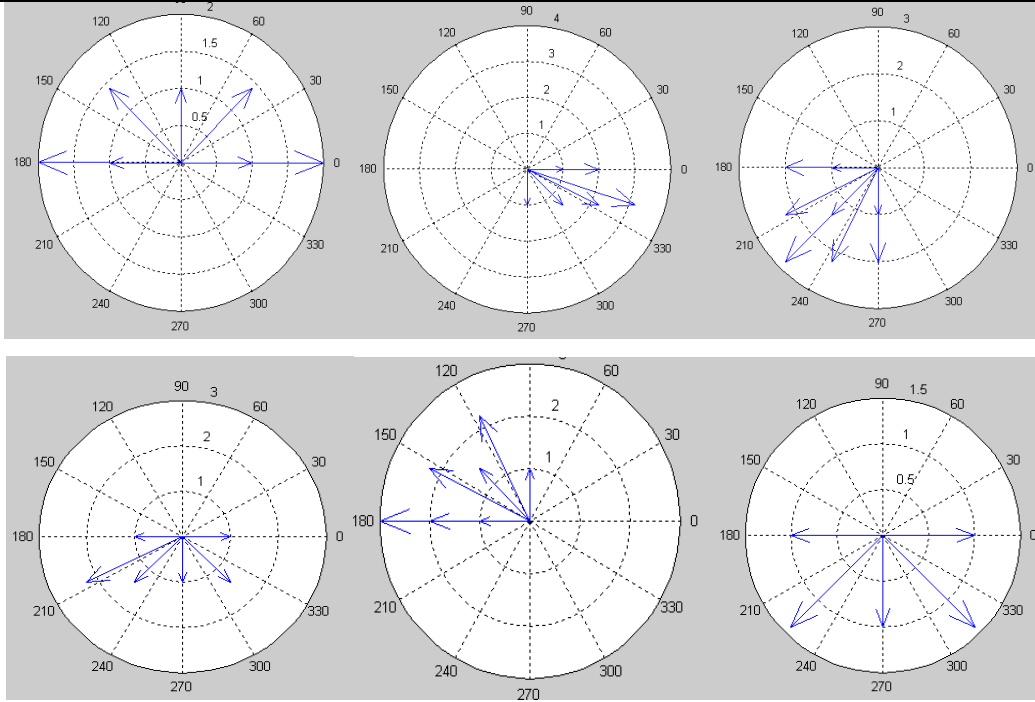


Figure 4. Average of the vector flow parameters for the six basis emotions in ROI.

Table 2. Recognition results of a section of AUs.

Action Units	Recognition Rate																	
	1	2	4	5	6	7	9	10	12	15	16	17	20	23	24	25	26	27
AU 1	89.4	20.3	13.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AU 2	29.3	79.8	11.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AU 4	12.4	21.4	88.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AU 5	0	0	0	86.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AU 6	0	0	0	0	81.9	14.9	0	0	0	0	0	0	0	0	0	0	0	0
AU 7	0	0	0	0	17.6	79.6	0	0	0	0	0	0	0	0	0	0	0	0
AU 9	0	0	0	0	0	0	79.9	0	0	0	0	0	0	0	0	0	0	0
AU 10	0	0	0	0	0	0	0	85.1	10.9	9.9	6.4	0	0	0	0	0	0	0
AU 12	0	0	0	0	0	0	0	16.9	79.9	14.5	8.1	0	0	0	0	0	0	0
AU 15	0	0	0	0	0	0	0	9.1	10.2	87.4	11.7	0	0	0	0	0	0	0
AU 16	0	0	0	0	0	0	0	7.2	9.2	92.1	3.1	0	0	0	0	0	0	0
AU 17	0	0	0	0	0	0	0	0	0	0	0	88.1	0	0	0	0	0	0
AU 20	0	0	0	0	0	0	0	0	0	0	5.1	7.2	85.1	6.1	9.8	0	0	0
AU 23	0	0	0	0	0	0	0	0	0	0	4.2	2.9	8.9	80.2	10.6	0	0	0
AU 24	0	0	0	0	0	0	0	0	0	0	5.2	4.1	9.9	10.9	81.1	0	0	0
AU 25	0	0	0	0	0	0	0	0	0	0	0	0	9.7	2.8	5.9	84.1	9.1	5.1
AU 26	0	0	0	0	0	0	0	0	0	0	0	0	3.8	5.3	9.4	12.7	81.1	6.1
AU 27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17.1	11	81

CONCLUSIONS AND FUTURE WORK

This paper describes a BN model for recognizing the “action units” of a face using video sequence images as input. The features were detected by using an optimal estimation optical

flow method coupled with a physical (muscle) model describing the facial structure. These muscle action patterns are then used for analysis, recognition, and synthesis of facial expressions. Our analysis tool consists of three main parts: 1) Region of Interest Selection, 2) Feature Extraction, and 3) Image Classification.

Bayesian Networks proved to be a powerful and flexible methodology for representing and computing probabilistic models in a stochastic process. In past decades' the optical flows has been used to either model muscle activities or estimate the displacements of feature points but in this thesis we find the nine interested regions (ROI) which contains the most complex motion by entropy maximum algorithm. Furthermore, the results were statistically analyzed by compass diagram to find out the major ranges of directions and velocities of vector flows in each ROI.

Finally, we apply the BN model to recognize single AUs and some important AU combinations. The average classification rate for the single AUs are between 80% and 90% and for the AU combinations are above 90%.

REFERENCES

- [1] P. Ekman and W. V. Friesen, The facial action coding system: A technique for measurement of facial movement. 1978.
- [2] I. A. Essa and A. P. Pentland, "Coding, analysis, interpretation, and recognition of facial expressions," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 9, pp. 757-763, 1997.
- [3] Yingli Tian, Takeo Kanade and Jeffrey F.Cohn, Recognizing Action Units for Facial Expression Analysis. In IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL 23, NO.2, FEBERUARY 2001.
- [4] T. Kanade, J. F. Cohn, Y. Tian, "Comprehensive database for facial expression analysis,". Proc. of the 4th IEEE Int. Con. On Automatic Face and Gestures Reco., France, 2000.
- [5] D. Datcu and L.J.M. Rothkrantz, Automatic recognition of facial expressions using Bayesian Belief Networks. In IEEE 0-7803-8566-7/04, 2004.
- [6] Jeffrey F. Cohn, James J. Lien and Adena J. Zlochower, Feature-Point Tracking by Optical Flow Discriminates Subtle Differences in Facial Expression. Department of Electrical Engineering University of Pittsburgh.
- [7] Agnieszka Oni'sko, Marek J. Druzdzel and HannaWasylyuk, Learning Bayesian Network Parameters from Small Data Sets: Application of Noisy-OR Gates. In Working Notes of the Workshop on 'Bayesian and Causal Networks: From Inference to Data Mining,' 12th European Conference on Artificial Intelligence (ECAI-2000), Berlin, Germany, August 2000.
- [8] J.L.Barron and N.A.Thacker, Tutorial: Computing 2D and 3D Optical Flow. www.tinavision.net, 2005.

[9] Jeffrey F. Cohn, Adena J. Zlochow, James Lien, and Takeo Kanade, Automated Face Analysis by Feature Point Tracking Has High Concurrent Validity with Manual FACS Coding. Department of Psychology, University of Pittsburgh and Robotics Institute, Carnegie Mellon University.

[10] Yaser Yacoob and Larry S. Davis, Recognizing Human Facial Expressions from Long Image Sequences Using Optical Flow. In IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 18, NO. 6, June 1996.

ABOUT THE AUTHOR

MSc Sunxioafan, Department of Mediamatica, Delft University of Technology

Prof.dr.dr.Leon Rothkrantz, SEWACO, Netherlands Defence University, Department of Mediamatica, Delft University of Technology, Phone: +31152787504, E-mail: L.J.M.Rothkrantz@tudelft.nl

Dr.ir.Pascal Wiggers, Department of Mediamatica, Delft University of Technology, Phone: +31152787504, E-mail: P.Wiggers@tudelft.nl