Spring 2012

An Efficient Boosted Classifier Tree-Based Feature Point Tracking System for Facial Expression Analysis

Adam Redd Livingston
Old Dominion University

Recommended Citation
Livingston, Adam R.. "An Efficient Boosted Classifier Tree-Based Feature Point Tracking System for Facial Expression Analysis" (2012). Doctor of Philosophy (PhD), Dissertation, Electrical & Computer Engineering, Old Dominion University, DOI: 10.25777/3me4-nr07
https://digitalcommons.odu.edu/ece_etds/98

AN EFFICIENT BOOSTED CLASSIFIER TREE-BASED FEATURE POINT TRACKING SYSTEM FOR FACIAL EXPRESSION ANALYSIS

by

Adam Redd Livingston
B.S. May 2004, Old Dominion University
M.S. May 2006, Old Dominion University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

ELECTRICAL AND COMPUTER ENGINEERING

OLD DOMINION UNIVERSITY

May 2012

Approved by:

Vijayan K. Asari (Director)

Jiang Li (Member)

Yuzhong Shen (Member)

Jim Leathrum (Member)

ABSTRACT

AN EFFICIENT BOOSTED CLASSIFIER TREE-BASED FEATURE POINT TRACKING SYSTEM FOR FACIAL EXPRESSION ANALYSIS

Adam Redd Livingston
Old Dominion University, 2012
Director: Dr. Vijayan K. Asari

The study of facial movement and expression has been a prominent area of research since the early work of Charles Darwin. The Facial Action Coding System (FACS), developed by Paul Ekman, introduced the first universal method of coding and measuring facial movement. Human-Computer Interaction seeks to make human interaction with computer systems more effective, easier, safer, and more seamless. Facial expression recognition can be broken down into three distinctive subsections: Facial Feature Localization, Facial Action Recognition, and Facial Expression Classification. The first and most important stage in any facial expression analysis system is the localization of key facial features. Localization must be accurate and efficient to ensure reliable tracking and leave time for computation and comparisons to learned facial models while maintaining real-time performance. Two possible methods for localizing facial features are discussed in this dissertation.

The Active Appearance Model is a statistical model describing an object's parameters through the use of both shape and texture models, resulting in appearance. Statistical model-based training for object recognition takes multiple instances of the object class of interest, or "positive" samples, and multiple "negative" samples, i.e., images that do not contain objects of interest. Viola and Jones present a highly robust real-time face detection system, and a statistically boosted "attentional" detection cascade composed of many weak feature detectors. A basic algorithm for the elimination of unnecessary sub-frames while using Viola-Jones face detection is presented to further reduce image search time.

A real-time emotion detection system is presented which is capable of identifying seven affective states (agreeing, concentrating, disagreeing, interested, thinking, unsure, and angry) from a near-infrared video stream. The Active Appearance Model is used to place 23 landmark points around key areas of the eyes, brows, and mouth. A prioritized binary decision tree then detects, based on the actions of these key points, if one of the seven emotional states occurs as frames pass. The completed system runs accurately and achieves a real-time frame rate of approximately 36 frames per second.

A novel facial feature localization technique utilizing a nested cascade classifier tree is proposed. A coarse-to-fine search is performed in which the regions of interest are defined by the response of Haar-like features comprising the cascade classifiers. The individual responses of the Haar-like features are also used to activate finer-level searches. A specially cropped training set derived from the Cohn-Kanade AU-Coded database is also developed and tested. Extensions of this research include further testing to verify the novel facial feature localization technique presented for a full 26-point face model, and implementation of a real-time intensity sensitive automated Facial Action Coding System.

This dissertation is dedicated to my father for his advice in navigating graduate study and life in general, my wife, Stephanie, for without her support and encouragement this document would never have been completed, and finally all of my family and friends who have kept me sane and grounded over the years.

ACKNOWLEDGMENTS

I have received a great deal of support and encouragement over the years from so many of those around me throughout my entire tenure as a graduate student. I first would like to thank my family and friends for the patience and unconditional support they have given me over years and years of difficult work. Particularly I would like to thank my father for the advice and support that could only have come from someone so much like myself who had already been through what I was working through in strikingly similar circumstances and situations. Furthermore, I would like to acknowledge the loving and patient support of my wife, Stephanie, without whom I am sure I would never have completed this document, defense, and degree.

For his continual support, guidance, and advice, I cannot thank Dr. Vijayan Asari enough. His patience, continued persistence, and advice both academic and otherwise have been a great help over the last 8 years. I thank Dr. Zia-ur Rahman, Dr. Jiang Li, Dr. Yuzhong Shen, and Dr. Jim Leathrum for the time and energy they dedicated to me, giving suggestions and advice on my research. I would also like to thank Laura Robb for her assistance in editing this document and generally helping make it more readable.

Acknowledgments are also in order for my fellow ODU Vision Lab students, Dr. Cort Tompkins, Dr. Praveen Sankaran, Dr. Jake Foytik, and Dr. Menna Youssef. Their help and friendship made my days working at ODU not seem like work. In particular I would like to thank Dr. Ming-Jung Seow and Dr. Hau Ngo for their early assistance to my research.

I would like to thanks my family and friends in particular, Justin O'Malley, Jason Liebler, Ryan Livingston, and Phil Coppersmith for their friendship, keeping me grounded, and making sure I stayed sane when I was working too hard. Lastly, I would like to thank Linda Marshall and Romina Samson for helping with the paper-work and answering all administrative questions about how to get things done in graduate school.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

Facial expression has been a prominent area of research for over one hundred years, since the early work of Charles Darwin [1]. The face serves as an important tool in conveying emotions, moods, traits, attitudes, personality, and many other signals vital to interpersonal communication. The manner in which facial expression is used to display and pass this information between individuals and how the internal emotional state is manifested in appearance has become an active area of research among psychologists. Pioneers of this research field include Paul Ekman, Wallace V. Friesen, Erika L. Rosenberg, and Jeffrey F. Cohn [2-5].

Two approaches have been taken towards the measurement of facial action: message judgment and sign-based measurement. In *message judgment,* an observer makes inferences about something underlying the facial action, such as emotion or personality. This most often refers to the detection of Ekman's six universal emotional states: anger, disgust, fear, joy, sadness, and surprise. However, the six universal emotional states were never intended to represent the full spectrum of human emotion. While providing a useful starting point, settling on the recognition and study of only these six states omits a large contingent of communicative, mood, and other emotions conveyed via facial expression. A second approach is that of *sign-based measurement,* in which inferences about the meaning of facial movement are left to higher-order decision making. Such systems treat the face as a vehicle for signal delivery. As a result they try to accurately follow and categorize facial movements. Interpretations as to the meaning of movements and combinations are left to the individual designing the automated system or conducting the experiment [3].

Before the work of Ekman and Friesen, the majority of research used individual and novel methods of measuring the appearance and action of the face. With the advent of the Facial Action Coding System (FACS), Ekman and Friesen sought to provide psychologists with a universal, observer-independent method of measuring facial movement and expression. To create FACS, experiments were conducted involving the electrical stimulation of individual facial muscles. The changes in appearance resulting from these stimulations were then grouped and categorized into prototypical facial action units (AUs). Each AU is defined as the change in appearance of the face and head as a result of the activation of one or a closely related group of facial muscles. The overall coding system can accurately detect and track even the finest facial movements [2].

FACS has gone through two revisions since its initial introduction in 1978, the most recent being in 2002. The following discussion of FACS and its various issues pertains to the 2002 version of the coding system. FACS is a comprehensive system which is psychometrically sound, can be readily learned by researchers, and has versatile utility to a variety of research applications. The current system contains nine AUs in the upper face, 18 AUs in the lower face, 11 AUs for representations of head position, nine AUs for tracking of eye position, four miscellaneous AUs, nine gross behaviors, and four supplementary codes. A detailed written description for the coding of each AU, as well as its intensity and how to manually create each AU, is included in the current FACS Manual [5]. Combination of AUs may occur in additive combinations in which the AU under examination is unaffected by other AUs which may be present. Non-additive combinations of AUs modify each other's appearance in such a way that their correlation must be taken into consideration [2].

The principle drawback of FACS is that it is designed as a manual system. Each frame of a video stream must be individually examined by a certified FACS coder for each of the AUs in consideration for the prospective experiment. As a result, FACS is a highly labor-intensive system, requiring even an expert coder to spend up to one hundred minutes to code one minute of video. Each AU is described by a written description which can leave some room for interpretation by the individual observer. This is especially the case when attempting to assign intensity measures. FACS includes descriptions for the categorization of five intensity levels (A-E). An intensity categorization of *A* indicates that the lowest level or only a trace of the prospective AU is present in the frame in question; an intensity categorization of *E* is assigned to the maximum strength of the action. The addition of intensity scoring to FACS 2002 adds the ability to track the dynamic motion (onset, apex, and offset) of a set of AUs as the frames progress [2]. The labor-intensive nature of FACS scoring and the subjectivity of assigning intensity to an AU can lead to a certain degree of inter-investigator ambiguity when scoring sequences.

The aforementioned issues become even more evident when spontaneous expression is under examination. In spontaneous facial behavior the presence of greater head motion and non-ideal conditions often leads to smaller sized, non-frontally aligned views and higher degrees of lighting invariance. Occlusions and movements which should not be taken in consideration (e.g. mouth movement during speech) may be present and have to be handled by the observer. Dynamics often play an important role in the detection and analysis of spontaneous facial behavior. Natural expressions tend to develop and fade slowly and smoothly with AUs peaking concurrently [2, 3]. Coding changes in intensity levels becomes more difficult as changes from frame to frame may be slight. Analysis of periodic motions and interpersonal patterns in triggered facial expressions is also extremely labor-intensive and difficult when performed using a manual system.

In [2] Cohn states that human-centric computing should seek to enable computers to unobtrusively perceive, understand, and respond appropriately to human emotion without the need for deliberate human input. To achieve this goal an iterative approach focusing on independent facial signals should be found in human-computer interaction which avoids solely focusing on strict emotion recognition. The need for computing systems is heavily stressed by the Cohn:

> *"Computing systems are needed that can automatically detect and dynamically model a wide range of multi-modal behavior from multiple persons, assess context, develop representations of individual differences, and formulate and test tentative hypotheses through the exchange of communicative signals."*

One possible extension of human-centric computing is the field of Human-Computer Interaction (HCI). HCI seeks to make human interaction with computer systems more effective, easier, safer, and more seamless. Graphic User Interfaces (GUIs) have been the primary style of user interaction with computers systems since they replaced command-line interfaces in the 1980s. WIMP-based (Windows, Icons, Menus, and Points) interfaces enable computer users to access the computer systems through visual representations directly controlled by the use of a keyboard and mouse. Though these systems are common and regularly understood by the major of computer users, they do have intrinsic weaknesses. GUI systems passively wait for the user to carry out tasks by means of a keyboard and mouse. Such systems are often limited to the input of single non-overlapping events. The way in which individuals utilize computers must be scaled to accommodate for a wider range of applications. For this reason interaction techniques which go beyond the tradition desktop metaphor must be examined [6].

The next stage in the advancement of computing devices and user interfaces involves the addition of more interactivity, responsiveness, and transparency. Efforts have been made towards the development of multi-modal, multi-media, multi-sensor user interfaces which emulate human-human communication with the long-term goal of creating an interface which makes use of natural means and expressive models of communication [7]. Multi-modal Perceptual User Interfaces (PUIs) have emerged as a potential candidate for the next interaction paradigm. PUIs are highly interactive, sensing and perceiving the world and taking actions based on goals and knowledge at various levels. Multi-modal interfaces make use of multiple perceptual modalities (e.g., sight, hearing, touch) from computer to user and vice versa. These interfaces move beyond the limited modalities and channels available with a keyboard, mouse, and monitor, to take advantage of a wider range of modalities, either sequentially or in parallel [8]. One important mode of input in which the HCI community is showing increasing interest is that of affective computing. Affective computing attempts to expand HCI by including emotional communication along with the appropriate means of handling affective information. Facial expressions are among the most natural means by which humans constantly and intuitively convey their affective state. Since the face can be passively monitored through the use of a digital camera, the facial expression is a convenient and fruitful mode to be incorporated into the development of HCI systems.

The goal of this dissertation is the development of an integral piece required for such a computing system. To detect and model facial behavior, a highly accurate method of locating key points and regions on an arbitrary dynamic face is necessary. Detection and tracking must be extremely efficient if a system is to incorporate higher-level modeling of facial structure and movement in real time. Fig. 1 shows a proposed block diagram of such a system. The proposed system can be broken down into three distinctive subsections: *Facial Feature Localization*, *Facial Action Recognition*, and *Facial Expression Classification*. Frames from a grayscale video stream containing a single prospective individual to be analyzed serve as the input to the novel system. For the purposes of development and testing of the algorithm presented, faces are assumed to be in a near frontal pose, have sufficient lighting, and be free of major occlusions (hands, scarves, sunglasses, etc.). Each frame is individually analyzed to locate the face and a set of key landmark points.

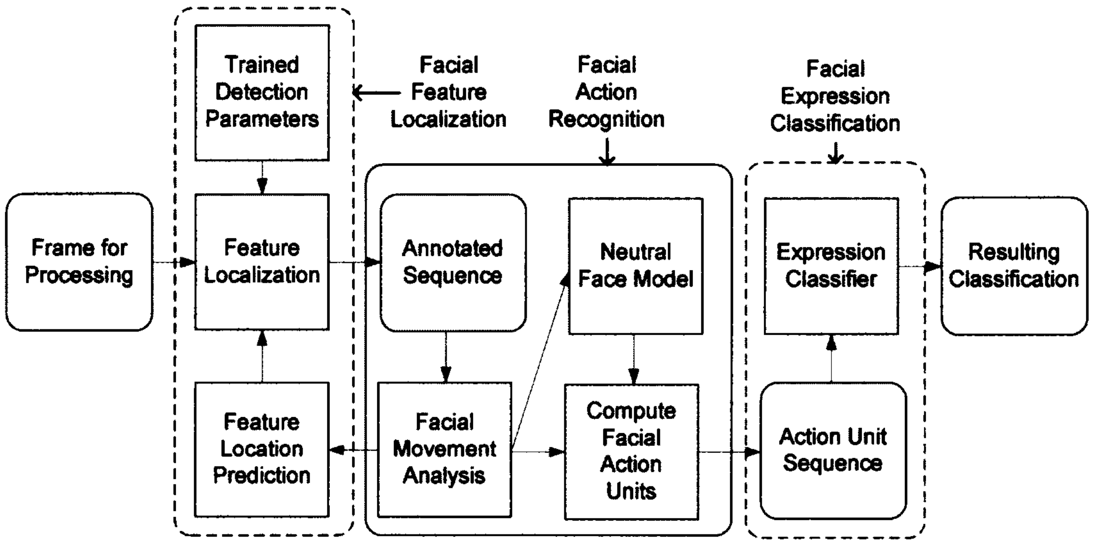

Trained Detection Parameters
Facial Feature Localization
Facial Action Recognition
Facial Expression Classification
Frame for Processing
Feature Localization
Annotated Sequence
Neutral Face Model
Expression Classifier
Resulting Classification
Feature Location Prediction
Facial Movement Analysis
Compute Facial Action Units
Action Unit Sequence

Fig. 1. Block diagram of facial expression analysis system.

The active appearance model technique for localizing facial features will be discussed in Chapter III. This will be followed by the discussion of a statistical feature-based object detection techniques in Chapters IV-V. A complete facial expression recognition system will be demonstrated in Chapter VI. Finally, Chapters VII and VIII present a proposed system designed to be a sign-based measurement system. Overall interpretation of facial action will be left to the prospective user of the system.

The remainder of this dissertation will be structured as follows.

Chapter II will present the current state of research related to the automation of facial action and expression analysis. A discussion of holistic and analytical approaches to facial action and expression recognition, as well as a hybrid approach to feature location, will be included.

Chapter III will discuss the Active Appearance Model and its use in feature localization in detail.

Chapter IV will discuss statistical object detection commonly used to locate facial regions. While Chapter V presents a novel approach to make these systems more efficient through the use of sub-frame reduction.

Chapter VIII will discuss an application of feature localization including a binary decision tree approach to human emotion detection for real-time HCI. This system makes use of Viola-Jones face detection, Active Appearance Modeling, and finally a binary decision tree to derive expression from facial movement.

Chapters VII and VIII will discuss design and implementation decisions as well as the development of a training set and testing set for locating facial features via a statistical feature-based approach.

Chapter IX will present conclusions, and options for future research.

II. Literature Review

Over the last two decades automated facial expression analysis has become an active area of research. Extensive surveys of early work in the field can be found in [9-11]. Recent work can be divided into two major approaches: analytic and holistic. Analytic approaches exploit flexible mathematical models to detect and track facial deformations and illumination changes. Discrete local facial features are used to locate key facial points around prominent facial features such as irises, eyelids, eyebrows, nostrils, mouths, etc. The locations of these facial points with respect to one another are used to gather information about the expression under examination. Active contour models, wavelet features, and rule-based approaches such as the FACS are included in this category.

Holistic approaches entail the use of gray-level template matching over the entire face template. Among the approaches included in this category are Neural Networks (NNs), linear discriminate analysis (LDA), principal component analysis (PCA), singular value decomposition (SVD), eigenfaces, and optical flow models. All of these approaches utilize the overall appearance of the entire face as a basis for judgments about expression [12]. Recently several hybrid approaches have been presented. Such approaches seek to take advantage of the precision and efficiency which can be gained through analytic or feature-based approaches as well as the universality of facial structure found in holistic facial model-based approaches. Recent examples from each of these categories will be presented in this section along with a discussion of the advantages and disadvantages of each approach.

A. Holistic Facial Expression Analysis

Holistic facial expression analysis is characterized by the examination of the face and its movement as a whole. The majority of holistic approaches to facial expression analysis can be broken into two stages: a tracking stage which follows changes in facial expression and a classification stage which makes some decision about the type of facial expression.

The most common method used for tracking expression movement through successive video frames is the use of Active Appearance Models (AAMs), which are discussed in Chapter III. AAMs attempt to learn the characteristics of both an object's shape and texture information. The result is a statistical model encompassing both the shape and texture variations of an object. PCA is used to normalize the training data. Linear models of both shape and texture are created, resulting in mean vectors, sets of orthogonal modes of variation, and model parameters. PCA is applied one more time to combine both shape and textural parameters into a final combined linear model. AAMs can then be used to generate a synthetic image which closely resembles a candidate object though an iterative search locating an optimal configuration of parameters. Key information about the location of facial feature points can be found by matching known points on the resulting generated synthetic facial image.

A recent example of this approach is found in the work of Beszedes and Culverhouse [13]. An AAM is trained to locate 58 landmark points around the eyes, brows, nose, mouth edge, and jaw line. Support Vector Machines (SVMs) with Gaussian radial basis functions (RBF) are then used to classify and input expressions as disgusted, happy, angry, surprised, or neutral at one of four intensity levels (0-3). An SVM creates a hyper-plain between two sets which maximizes the margin between two data sets. The AAM-SVM system described was tested on a set of 12 individuals displaying four facial expressions at four different intensity levels and three different scaling levels. The results are compared to human evaluations in which 24 individuals were given the same recognition task. The automated system slightly outperforms the humans at the task of emotion recognition with an accuracy of 70.31% to 68.75% respectively. However, the human subject slightly outperforms the AAM-SVM when it comes to accurately detecting the level of a correctly identified emotion with an accuracy of 53.26% to 50.00% respectively.

The work of Lucey, et al. [14] seeks to test various AAM-derived face representations with different combinations of classifiers and normalization techniques. A 2D shape model based on a wire frame model, a 2D shape-free texture model, and a 3D model based on a triangular mesh are all tracked using an AAM person-specific model. LDA with nearest-neighbor classification and SVM-RBF classification are applied to the task of detecting four AUs related to the movement of the eyes and brows. Normalization based on the whole face as well as normalization based solely around the eye region is also tested. Results on the Rutgers University2 database found that there is no benefit to the use of 3D shape representation and that normalization around the region of interest, in this case the eyes, significantly improved the performance. An improvement in performance is also seen when neutral expression information is used.

Another possible approach to classification of facial action involves the use of NNs. Kuilenburg, et al. [15] posed a method of identifying seven emotional states and 27 AUs using a combination of AAM face tracking and NNs. The appearance vector from the AAM tracking is fed into two three-layer feed-forward NNs. The first network, with dimensions of $94 \times 15 \times 7$, is trained to translate the appearance vector of length 94 into the emotions happy, angry, sad, surprised, scared, disgusted, and neutral with 15 nodes contained in the hidden intermediate layer. An overall accuracy of 89% is achieved on the Karolinaska dataset. The second network, with dimensions of $94 \times 20 \times 5$, is able to accurately detect 15 AUs with an average accuracy of 86%. Another approach utilizing feed-forward NNs is developed by Ma and Khorasani [16]. However, in their approach the 2D discrete cosine transform (DCT) coefficients of a difference image are used as an input to a $144 \times 6 \times 4$ feed-forward network. A recognition rate of 93.75% is achieved over a database containing neutral, smile, anger, sadness, and surprise expressions.

Dornaika and Davoine [17] present a particle-based framework for tracking 3D motion of inner facial features using a statistical facial appearance model. An adaptive observation model and an adaptive transition model are used in combination with a 3D wire frame. The basic principle of this technique relies on the hypothesis that if a 3D wire frame model corresponds to the actual 3D face then the associated shape and texture will correspond to a facial texture. The resulting face tracker accurately follows facial motion after a manual initialization of the key wire frame points. Performance is successful in cases involving significant occlusions as well.

Kotsia and Pitas [18] present a facial expression recognition system using a Candide grid and grid tracking based on deformable models. Key points are manually placed on the first frame of an image sequence. A KLT grid-tracking algorithm based on deformable face models is used to track the motion of the face through the image sequence. The frame with the greatest facial expression intensity is then used to calculate grid node displacements. Two different types of grids are used to track facial motion and identify Ekman's six universal emotional states. The first contains 104 points covering the entire facial region while the second is reduced to 62 key points around facial features. The 62-point facial grid achieves the best performance over the Cohn-Kanade dataset using a six-way-multiclass SVM for classification with an accuracy rate of 99.7%. A second approach is attempted in which 17 AUs are detected using an individual two-class SVM for each AU. An overall detection rate of 94.5% is achieved over all AUs using the 62-point facial grid. When applying Ekman's FACS rules for the six emotions to the 17 AUs a detection rate of 95.15% is achieved. This study finds that better performance can be achieved by accurately following a reduced set of facial points around key facial features.

Anderson and McOwan [19] work towards the development of a real-time, fully automated system for the recognition of six universal emotional states. Faces as a whole are located and tracked using a ratio template matching algorithm searching for the average luminance of using a spatial face model enhanced by including biological proportions universal to all faces. An optical flow algorithm using a multi-channel gradient model operating in three dimensions, two spatial and one time, is used to track facial movement. Six *1v.All* SVM classifiers with a Gaussian RBF are used to detect the emotional state in each frame. An 80.52% accuracy rate is achieved using this approach on the CMU-Pittsburg AU-coded facial expression database. By reducing the frame rate and image size, utilizing ratios of average motion in the tracking process, and implementing the optical flow tracking using a Matrox Genesis DSP board, a frame rate of four frames per second is achieved.

A unique approach to facial expression analysis is developed by Lee and Elgammal [20]. Their approach attempts to model the sequence of a facial expression as a one-dimensional closed manifold beginning and ending with a neutral expression. Nonlinear decomposable generative models are learned which can generate dynamic facial appearances for different individuals and expressions. Results on tests over seven individuals from the Cohn-Kanade database over six expression sequences lead to a 40% expression recognition rate.

Facial expression analysis approaches of this type often provide accurate tracking results. However, achieving these results often requires complex iterative models which focus on the movement of the face as a whole. Due to the iterative nature of these tracking schemes, real-time performance becomes extremely difficult to achieve. In most cases techniques bypass the detection of individual AUs, instead proceeding directly to some broad categorization based on the emotional state. Several studies hint at some major concepts such as the use of local modeling and neutral expressions to increase performance.

B. Analytic Facial Expression Analysis

Analytic approaches to facial expression analysis are characterized by the motion of landmark facial features. Bassili and Bruce performed an early study commonly referred to as the Johansson's point-light display experiment which serves as the starting point for the majority of analytic approaches. In their experiments white marks were placed randomly on the faces of individuals. Test subjects were then shown a monitor containing only the white points. Viewing only the white points, it was quickly evident to the test subjects that faces were being observed and that demonstrated expressions could be identified. A point-based face model was then developed in which key points are placed around the eyes, brows, nose, mouth, chin, and jaw [21].

Analytic facial expression analysis usually contains three main steps. The facial landmark points must first be identified and extracted. The location of the points in relation to one another or their movement over frames is analyzed to extract information about the presence of any facial movement. Lastly, a detector or classifier is used to identify either an expression or facial movement depending on the target application of the system.

One of the most active research groups in this area of expression analysis is led by Pantic [22]. Their earlier work involves analysis of facial expression from static face images. Images are captured from a special-head mounted camera setup which simultaneously captures both a frontal and profile image of a subject. A HSV color space skin segmentation is used to identify the facial region in each image. Multiple redundant detectors are used for localization of the eyes, brows, nose, mouth, and chin. As a result of the best detector results, 18 frontal face landmark points are placed. Analysis of the contour of the profile image is used to place 10 profile points at locations such as the tip of the nose, chin, lips, etc. Mid-level features are then drawn from a specially chosen combination of the 28 landmark points based on observed accuracy. A rule-based system is then used to translate mid-level features into one of 32 AUs. The main application of this system is in the development of a tool which can assist psychologists in the study of facial expressions.

Work on improving the analysis of profile view is presented in [23]. The objective of this work is to expand the range of automatically observable human facial behavior by adding the recognition of the temporal characteristics of AUs and increasing the number of AUs detectable from the profile view. The initial localization of 15 facial points is not addressed in this work. A particle filtering technique which maintains a particle-based representation derived from an a posteriori probability model is used to track the motion of the 15 landmark points throughout the image sequence. The movement of each of these points, both up/down and towards/away from the head, is fed to a rule-based detection system. The result is the detection of 27 AUs with the additional label of *onset*, *apex*, or *offset* describing the temporal state of each AU. A recognition rate of 86.6% is achieved over 119 profile test images taken from the same head-mounted rig described above.

Vukadinovic and Pantic [24] further work to automate the placement of 20 facial feature points on any frontal face image using Gabor feature-based boosted classifiers. The face is first detected using a modified version of the Viola-Jones face detection cascade. Regions of interest (ROI) are then found using vertical and horizontal histograms. Each specific ROI is then searched by scanning a 13×13 search window across the region. During training stage, a positive set of 13×13 images of each feature point is fed through a bank of Gabor wavelet filters. Eight spatial orientations at six frequencies are computed at each point in the search window, yielding 8,281 total features per search window. GentleBoost is then used to select the 300 features which best represent each feature point. AdaBoost yields a discrete response. The GentleBoost algorithm results in an analog result for which a large positive classifier response corresponds to a high confidence that the area beneath the search window is the desired feature point. An accuracy of 93% of the facial points is within 10% of the distance between the inner corners of the eyes over the Cohn-Kanade database.

Valstar and Pantic [25] use the automatic feature point localization by Gabor feature boosted classifiers as the front end for a temporal facial AU detector. Once the 20 points are initially located, the same particle filtering with the factorized likelihoods technique from [23] is used to track the facial points between successive frames. Facial movement features are then calculated for each frame. These features include the offsets from the neutral frame in both the x and y directions, the inter-point distances within the current frame, and the differences between the inter-point distances as compared to the neutral frame. The facial movement features are then ordered for each AU using the GentleBoost algorithm. A two-class SVM is created using the ordered facial movement features. Correctly detected AUs are then classified further in the temporal stage of *onset*, *offset*, *apex*, or *neutral* by a set of six *1v1* SVMs with the winner chosen by a voting scheme. This process is repeated, creating 15 detection networks, one for each AU being detected, with each network containing seven SVMs. Over 15 AUs from the Cohn-

Kanade database an AU recognition rate of 90.2% is achieved. When the correct AU is selected, its temporal classification is correct 95.0% of the time.

Another research group taking a similar approach is that of Bartlett and Littlewort with an emphasis on spontaneous behavior and expressions. A fully automatic facial action recognition system with the ability to detect 20 AUs with a detailed analog intensity measure is presented in [26]. A solid background on the issues related to the differences and challenges related to spontaneous versus non-spontaneous expression is presented along with a discussion about the need for spontaneous databases. A modified Viola-Jones face detection cascade is used to identify faces. The entire face is then scaled down to a 48×48 window and is fed through a Gabor filter bank with eight orientations and nine spatial frequencies. For each AU AdaBoost is run on a set of positive and negative training images creating an ordered list of the best features. The GentleBoost learning algorithm is then used to create a classifier for each AU using its 200 best features. The margin of each classifier closely correlates with the intensity of the AU. As a result very detailed information about an expression at any moment in time can be found and used for study. An average recognition rate of 91% was found across 20 AUs over two posed databases (Cohn-Kanade, Ekman-Hager). On a spontaneous database a recognition rate of 93% was found.

Bartlett extends the same technique by adding analysis of multiple AU occurrences and a study as to the effect of image compression on the technique in [27]. Littlewort and Bartlett also applied the same feature detection technique using an SVM voting classifier to the detection of Ekman's six universal states in addition to a neutral expression. The problem was treated as a seven-way forced choice. SVM classifiers are trained for each possible partition of the seven classes of expression. For each classifier the best 900 features are selected by AdaBoost ordering and then used to train the SVM. A real-time frame rate of 24 frames per second is achieved while maintaining a 93% accuracy rate [28].

Several researchers in the field explore complex methods of expression and facial movement classification. Wang, et al. [29] classify seven types of expressions using a histogram-based continuous multi-class AdaBoost approach. A 2D lookup table is used to assign a real value to an input image based on its response to a subset of Haar-like features. An accuracy of 92.4% is achieved on the JAFFE database requiring only 0.11 ms of analysis per image. Aleksic and Katsaggelos [30] utilize the facial animation parameters supported by MPEG-4 and a method of training a set of multi-stream Hidden Markov Models (HMMs) to detect seven types of expression. The facial action parameters of the brows and lips are used to train two separate single stream HMMs. The two HMMs are then combined with weightings set proportionally to the information contained in the two streams. A 93.66% detection rate is achieved over the Cohn-Kanade database.

Zhang [31] presents a method based on a three-layer Dynamic Bayesian Network (DBN). Active infrared is used to detect and normalize the facial region. Twenty-six facial feature points are then placed by a detector derived from 18 Gabor wavelet coefficients and tracked by Kalman filtering. Collections of these feature points comprise the feature layer. An AU layer connects the feature layer to the classification layer. Training samples are used to parameterize the weights between the layers. An HMM is used to link the DBNs for successive time slices and monitor the dynamics in the change of expressions. The system actively selects the most informative visual cues presented by the face under analysis allowing for accurate and robust recognition of spontaneous facial expression.

The use of DBNs in expression recognition is pioneered by the work of Kaliouby [32]. An overview of a mind-reading machine is offered in [33]. A multilayer DBN is used to classify the complex mental states of *concentration*, *disagreement*, *interest*, *thinking*, and *unsure*. A color-based analysis is combined with feature location to extract facial features and estimate head pose. Twenty-four facial feature points are used to detect head motion and AUs while HMMs translate movements into head and facial displays. A three-layer DBN then detects the six complex mental states with an accuracy rate of 87.4%. Kaliouby tests the generalization of such a DBN system against the performance of human test subjects. Among the 18 human participants a mean accuracy of 53.03% is found with only two individuals categorizing expressions at a rate above 85%. The automated system achieves a mean detection rate of 63.5%, which falls in the top 5.6% among the human trial results [34].

The work of Shan, et al. [35] takes a unique approach to feature extraction through the use of an alternate image representation on low-resolution images. Local binary patterns (LBPs) are formed by performing a binary threshold operation on pixels surrounding a center pixel by the value of the center pixel. The clockwise values of the thresholded pixels then form an eight-digit binary code. The face under analysis is divided into regions and a histogram of binary codes is formed for each region. Two-class SVM classifiers with a simple voting scheme are used to classify Ekman's six universal emotional states over the Cohn-Kanade dataset. Accuracy of 92.1% and 75.8%

is found at resolutions of 110×150 and 14×19 respectively. LBP features can be derived rapidly on a single scan of the raw image while still retaining enough facial information in a compact representation. Feng, et al. [36] use a similar feature extraction approach with a linear programming classification technique. Linear programming attempts to generate a plane that minimizes an average sum of misclassified points belonging to two disjointed sets. Twenty-one two-class linear programming classifiers are created to classify Ekman's six universal emotional states. 93.8% accuracy is achieved over the JAFFE database.

Analytic facial expression analysis techniques achieve a high level of accuracy in most cases. This results from the use of specialized detectors, mostly built from Gabor feature banks and in some cases LBPs. Gabor feature banks have the advantage of a high level of description but may be time-consuming to repeatedly compute. LBPs can be quickly computed but lack the resolution offered by Gabor features. Face detection and feature point location are always held as two separate processes in all current approaches from this class of algorithms. As a result, feature location must rely on separate processes to define ROI for searches.

C. *Hybrid Facial Motion Tracking*

Hybrid approaches to facial motion tracking attempt to combine the best properties of the global nature of holistic facial expression analysis with the specificity in location that can be achieved through analytical facial expression analysis. The work of Cristinacce and Cootes pioneers this specific approach in a series of three papers.

In the first paper, [37], a set of Viola-Jones AdaBoost detection cascades is trained to locate each individual feature point over the face. However, it has been found that there is insufficient local structure to accurately locate feature points using a global search of the face. There are simply too many false positives for this approach to be practical. Performance is improved by using a global shape constraint with a least square fit algorithm. Only eye and mouth corners are located. A subset of the facial locations with the best responses is taken for each detector. Faces are then constructed out of each possible combination of these points and tested against a learned shape model. The combination which best fits the shape model is chosen as the winning facial layout. Mean error is less than 15% of the inter-ocular distance for 90% of faces in the XM2VTS dataset.

Cristinacce and Cootes further refine their facial feature detection and tracking system by utilizing a set of feature templates with a shape-constrained search technique. The face is first detected using a Viola-Jones face detection cascade. Initial point locations are placed based on average location derived from a training set. Rectangular regions are extracted around each point and compared using a nearest-neighbor search with a set of templates. An image search is performed with each winning template and the locations with the best response are used to optimize a shape model. Landmark points are then updated. The search procedure is repeated until there is no movement in the points. Test over the BIOID and XM2VTS datasets result in 96% of the test images having a mean error of less than 15% of the inter-ocular distance [38]. The search algorithm is further enhanced by the use of constrained local models (CLMs) taking both shape and textural appearance into account. Shape and appearance of rectangular regions around known feature points of training images is combined into local models by PCA. Extracted rectangular regions are compared to templates generated by the CLM with the best template selected by a similarity transform. The remainder of the search algorithm is the same as described above. 95% and 92% of test images had a mean error less than 20% of the inter-ocular distance in the BIOID and XM2VTS datasets, respectively [39]. Both of the previously described methods achieve a real-time performance of four frames per second during the initial point location and 25 frames per second during tracking on 384×286 images on a Pentium IV 3Ghz processor [38, 39].

Falzenszwalb pursues a tracking approached based on pictorial structures for object recognition. Each object is represented by a collection of parts arranged in a deformable configuration. The appearance of each object part is modeled separately and linked to one another by spring-like connections. Models of individual object components and their spatial relations to one another are learned through a training set. An implementation of a tracking system which successfully detects and follows the eyes nose, and corners of the mouth is demonstrated in [40].

Wang, Zou, and Sun [41] observe that in the majority of facial expression analysis algorithms, face detection is either assumed or performed by Viola-Jones face detection as a precursor to feature location. They seek to develop a probabilistic face model-correlating classifier response found during face detection to known facial feature points. A Viola-Jones boosted case detector, implemented using OpenCV, is trained using the IMM Face Database as a positive set and the Caltech 101 Object Categories as a negative set. A 17-point face model is learned from correlations between feature points occurring within the rectangular regions of the Haar-like features. A mean error of less than 14%

of the inter-ocular distance is achieved for 88% of the faces in BIOID dataset. A significant speed up is observed over AAM feature tracking systems and the shape-constrained facial feature detectors developed by Cristinacce while maintaining a similar accuracy.

D. Emotion Classification Systems

Cohen, et al. [42] developed an early system to recognize human facial expression from video sequences using a multilevel HMM while increasing the discrimination power between different classes of basic emotions. Their system trained a single HMM to detect each of the emotions (happy, angry, surprise, disgust, fear, and sadness) based on the continuous motion of facial AUs. A training set of five individuals performing six emotional states was converted into a sequence of 12 AU-like measurements. When tested in a person-dependent manner, the multi-level HMM approach achieved an 82.45% recognition rate. When trained and tested as a person-independent system, only a 58% recognition rate was achieved. One of the main weaknesses of this study lies in the fact that the video sequence had to be manually annotated for the 12 AUs it took as input.

Esau, et al. [43] improved on this concept through their development of the fuzzy video-based emotion recognition system, VISBER. The main innovation offered by this system is the ability to detect blended emotional states with varying intensities and automatically adapt to the characteristics of an individual's face during a short training phase. Six angles constructed from 12 fiducial points are mapped into the categories small, medium, and large. Depending on this combination experimentally devised rules are used to perform classification. When given 200 quality examples of an individual displaying neutral, anger, fear, happiness, and sadness expressions, an average recognition rate of 72% was achieved. Happiness and sadness showed somewhat lower recognition results due to inaccurate placement of mouth feature points.

Dhall, et al. [44] implemented an emotion recognition system which made use of a pyramid of histogram of gradients (PHOG) and local phase quantization features. Face tracking was accomplished using a constrained local model where landmarks are located by utilizing an ensemble of local feature detectors which are then combined by enforcing a prior of their joint motion. Shape features were found by PHOG, which counts occurrences of gradient orientation in localized portions of an image. Appearance feature extraction was accomplished through Local Phase Quantization (LPQ) which is based on computing a short-term fourier transform in a local image window. A recognition rate of 72.4% was achieved in a test run over the GEMEP-FERA dataset.

Another trend which has arisen in the field of emotion detection is the use of auditory information in conjunction with video image sequences to improve recognition rates. Examples of approaches exploring various methods of fusing both modes of information into a single classification result can be found in [45-47].

The approaches described in this chapter have both strong and weak points. They do achieve accurate performance and real-time performance in most cases, usually by combining a local approach within a globally-derived shape structure. The principal weakness of all the above techniques is their focus on only a single level of resolution.

III. AAM Feature Localization

Active Appearance Models (AAM) were first developed by Cootes, Edwards, and Taylor in 1998 [48]. AAM is a statistical model describing an object's parameters through the use of both shape and texture models, resulting in appearance. When considering facial feature localization, the shape corresponds to the outlining contours plus some of the inner edges of the face. The appearance of the face is defined as the texture in a shape-free space. The facial model becomes "active" by learning its statistical borders of representation from a set of annotated images in a one-time training session. Similarities in the model are optimized offline, significantly speeding up later convergence in the underlying high-dimensional space of the model.

A. Active Appearance Model

The AAM is a generalization of the widely used Active Shape Model (ASM). The AAM approach makes use of image texture for matching, while the ASM relies solely on near modeled edges. An AAM contains a statistical model of the shape and grey-level appearance of the object which can be generalized to almost any valid example. This approach is especially appropriate for facial feature localization since a key initial assumption is that each face will contain several key features which must be localized. The process of matching the facial template to a candidate facial image involves finding model parameters which minimize the difference between the image and a synthesized model example projected on the candidate image. Due to the large number of parameters, accurate feature placement becomes a difficult problem.

Displacing each model parameter from its correct location induces a pattern in the residuals produced. The AAM learns a linear model of the relationship between parameter displacements and the induced residuals during the training phase of the algorithm. During the search for an optimal set of matching parameters residuals are measured and used to modify the current parameters toward a better fit. With this approach it is possible to find a good overall match in a few iterations even from a poor starting estimate. Fig. 2 shows frames of the use of an AAM to match a face template to a candidate face image. As the number of iterations increases, the template face become more closely mapped to the candidate face and the projection errors are minimized.

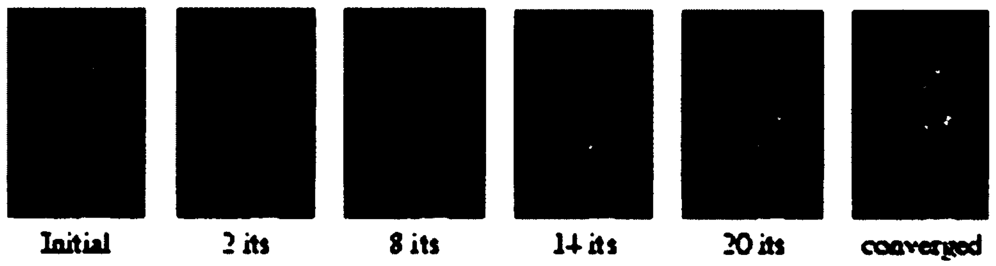


Fig. 2 Active appearance model example.

The remainder of this chapter will discuss in further detail the creation of a system for matching a face template to an unknown facial image. The formation of shape and texture models will be discussed. Next the combination of these two models will be presented. Finally, an AAM matching algorithm will be given.

B. Shape Formation

An AAM represents planar shapes as a finite set of landmarks distributed over a facial region. The representation used for a single *n*-point shape is given as

$$x = [x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n]^T \tag{1}$$

PCA is used to reduce the redundancies which are inherently present in multivariate data such as shapes. In the application of facial feature mapping a shape of n points is considered as single observation, x_i, in a *2n* dimensional space.

Shape-based PCA is conducted by performing an eigen-analysis of the covariance matrix of the shapes aligned with respect to position, scale, and rotation after a Procrustes analysis. New shape instances are synthesized by deforming the mean shape, $\bar{x}$, using a linear combination, b_s, of the eigenvectors of the covariance matrix, Φ_s:

$$x = \bar{x} + \Phi_s b_s \tag{2}$$

Points of the shape are transformed into a modal representation. Modes are ordered according to the percentage of variation between the template and candidate landmark points. To regularize the solution space and improve performance, modes are included until the accumulated variation explained by the model is above a certain threshold (e.g. 95%). In other words, the model is built up until at least 95% of the possible variation between the template landmark locations and a training set can be accommodated by the shape matching model.

C. *Texture Formation*

Texture is defined as the pixel intensities across the object in question, possibly after the appropriate normalization. For m samples over the object's surface the texture is represented as:

$$g = [g_1, g_2, \ldots, g_m]^T \tag{3}$$

In the case of shape representation, the data acquisition is straightforward because the landmarks in the shape vector are the shape data. A consistent method for collecting texture information between landmarks is needed. A piece-wise affine warp based on the Delauney triangulation of the mean shape is used to sample each training image and gather texture data.

Normalization of the g-vector set is performed to void the influence from global linear changes in pixel intensities. PCA analysis is then applied to the set of textures in the same manner which was applied to the shape vector. The compact representation which is derived to deform the texture in a manner similar to that which occurs in the training set is given by:

$$g = \bar{g} + \Phi_g b_g \tag{4}$$

In this equation g is the mean texture; Φ_g denotes the eigenvectors of the covariance matrix; and b_g is the set of texture deformation parameters.

D. *Combined Model Formation*

With a representation established for both the shape and texture models developed, a method must be utilized to further compact these models. To accomplish this task a third PCA is performed on the shape and texture PCA scores over the training set, b, to obtain combined model parameters, c:

$$b = Qc \tag{5}$$

The PCA scores are directly obtained due to the linear nature of the model:

$$b = \begin{bmatrix} W_s b_s \\ b_s \end{bmatrix} = \begin{bmatrix} W_s \Phi_s^T (x - \bar{x}) \\ \Phi_g^T (g - \bar{g}) \end{bmatrix} \tag{6}$$

A suitable weighting between pixel distances and pixel intensities is obtained through the diagonal matrix, W_s.

Now a complete model instance including shape, x , and texture, g, is generated using the combined model parameters, c:

$$\begin{aligned} x &= \bar{x} + \Phi_s W_s^{-1} Q_s c \\ g &= \bar{g} + \Phi_g Q_g c \end{aligned} \tag{7}$$

The rank of Q correlates to the compression of the model parameters and will never exceed the number of examples in the training set.

E. AAM Matching

Once parameter derivatives for the model, texture, and pose are learned from a training set, matrices R_c, R_t, and R_p are constructed for use in a tracking algorithm. An iterative approach is used which gradually matches the model patch to the target image. The first step is to calculate the residual between the target image and the model patch and the intensity error of the current mapping for the current iteration, i.

$$\begin{aligned}\delta g_i &= g_s - g_m \\ E_0 &= |\delta g_i|^2 = |g_s - g_m|^2\end{aligned} \tag{8}$$

Next, using precompiled gradient matrices, the model parameter, pose, and texture are updated.

$$\begin{aligned}\delta c &= R_c \delta g_i \\ \delta p &= R_p \delta g_i \\ \delta t &= R_t \delta g_i\end{aligned} \tag{9}$$

An update parameter k is set equal to one and new estimates for the model parameters are computed for the model, pose, and texture.

$$\begin{aligned}c_{i+1} &= c_i - k\delta c \\ p_{i+1} &= p_i - k\delta p \\ t_{i+1} &= t_i - k\delta t\end{aligned} \tag{10}$$

A new model is then calculated based on the updated model parameters. A new residual is then calculated and the intensity error is calculated. If $E_{i+1} < E_i$, then c_i, p_i, and t_i are selected as the new parameter vectors. If this is not the case, then $k = 1.5$, $k = 0.5$, $k = 0.25$, and $k = 0.125$ are tested. This process is repeated until no further improvements are achieved or a fixed number of iterations are reached.

IV. Statistical Feature-based Object Detection

Statistical model-based training takes multiple instances of the object class of interest, or "positive" samples, and multiple "negative" samples, i.e., images that do not contain objects of interest. Positive and negative samples together make a training set. During training, different features are extracted from the training samples and distinctive features that can be used to classify the object are selected. This information is "compressed" into the statistical model parameters. If the trained classifier does not detect an object (misses the object) or mistakenly detects the absent object (i.e., gives a false alarm), it is easy to make an adjustment by adding the corresponding positive or negative samples to the training set.

The goal of object detection in digital images involves the detection of an image sub-region containing groupings of pixels which belong to a specific class of objects while eliminating any sub-region not containing the desired object. A major breakthrough in real-time object detection is presented by Viola and Jones in [49, 50]. The Viola-Jones real-time face detection framework has become an industry standard and is utilized as the front end of numerous face identification and verification systems as well as emotion analysis systems.

Viola and Jones present three major innovations which, when combined, yield a highly robust real-time face detection system. The first involves the use of an integral image representation which allows for the instantaneous calculation of the sum of pixels in any rectangular area of an image through a weighted sum of its corner values. An integral image representation can be calculated in a single scan of a frame or image to be analyzed. Rectangular Haar-like features (Fig. 3) can then be computed at any scale or image location in constant time through a linear sum of the corner positions of the feature. Haar-like features are capable of robustly representing generalized edges, lines, and regional characteristics of an object over a training set.

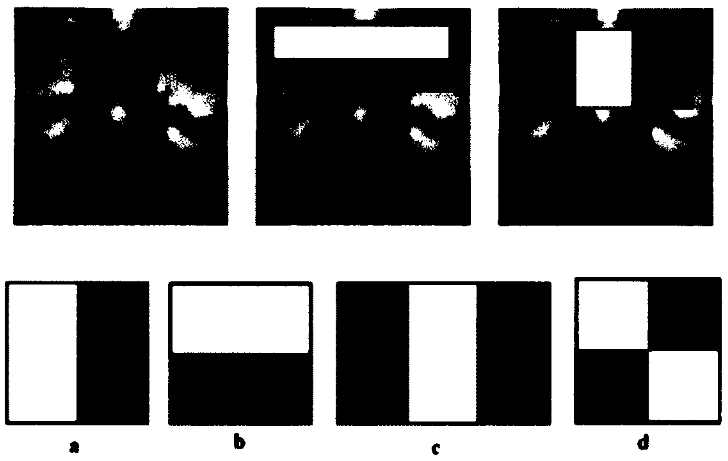


Fig. 3 Haar-like features.

The second element is the application of a statistical boosting algorithm to select appropriate features that can form a template to model human face variation [51]. A training set containing positive and negative examples is used to find those features which best partition the two sets. A boosting coefficient proportional to the classification performance is assigned to the selected feature. Training images are reweighed such that misclassified images receive greater consideration in the selection of the next feature. More boosted features are required for the correct classification of a difficult image.

An "attentional" cascade of classifiers speeds up the search by exploiting this property of boosted classifiers. By quickly eliminating unlikely face regions, computationally complex classifiers are reserved for those sub-frames which closely resemble faces. Each stage in the cascade is trained to be progressively stronger. Only face sub-frames must be evaluated by all features in the cascade. The vast majority of sub-frames are eliminated in the first several levels of the cascade, saving significant computational time.

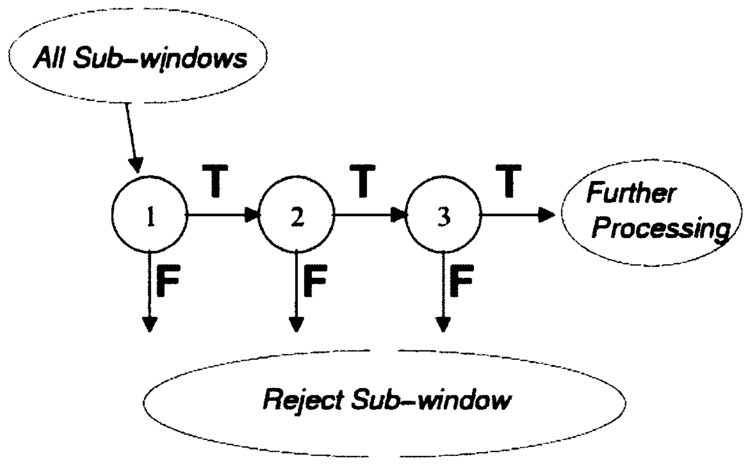


Fig. 4 Cascaded classifier.

The remainder of this chapter will discuss first the features chosen for construction of a set of weak classifiers and their calculation. Next the manner in which these features are combined for the creation of a generic classifier-based detector using both AdaBoost and Gentleboost statistical training methods is discussed. The construction and training of a cascade detector will then be introduced.

A. Haar-like Features Value Calculation

A large and varied feature set is required from which to draw weak classifiers to serve as the build blocks for a strong cascaded classifier. Viola and Jones introduce a method relying on four types of simple Haar-like features constructed out of combinations of upright rectangles. The value of each feature is computed by taking the difference of sums of pixels contained within the adjacent rectangular regions. The types of Haar-like features to be used in the proposed technique are shown in Fig. 5.

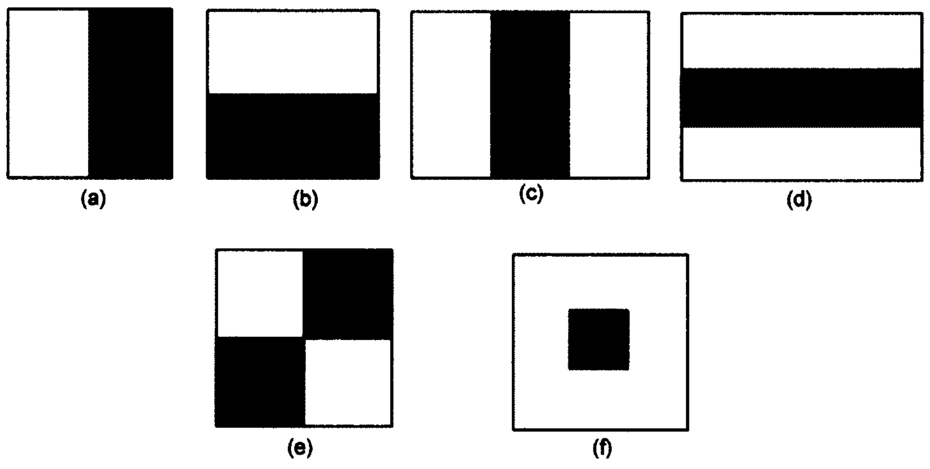


Fig. 5 Haar-like features.

Features (a) and (b) are used to detect horizontal and vertical edge-like image segments. Combinations of these simple edge detectors are combined to create vertical, horizontal, and diagonal line detectors as well as a center surround feature which detects symmetrically centered objects such as points, circle, and objects differing from their background [52]. Each feature was shifted and scaled in both the *x* and *y* directions within a 24×24 detector to create a set of features, *H*.

$$H = \{h(tp, x_o, y_o, x_l, y_l)\} \tag{11}$$

Each individual feature, $h(tp, x_o, y_o, x_l, y_l)$, is defined by five parameters. The first parameter is the type of Haar-like feature, *tp*, which references one of the six features types shown in Fig. 5. Next is the offsets of the top-left corner of the feature with respect to the top-left corner of the image in both the x and y directions. Finally, there is the width and the height of the feature as defined by x_l and y_l. For a feature to be valid and included in this set several conditions must be met. The entire feature must be contained within the dimensions of the detector.

By using integral image representations the sum of any rectangular region can be computed in constant time through a weighted summation of its corner points. The summed area table contains the sum of all pixels above and to the left of any given point in an image and can be computed in a single scan of the input image, $I(x, y)$.

$$S(x,y) = \sum_{x'=0}^{x} \sum_{y'=0}^{y} I(x',y') = \sum_{x'=0}^{x} \sum_{y'=0}^{(y-1)} I(x',y') + \sum_{x'=0}^{(x-1)} I(x',y) + I(x,y) \tag{12}$$

$$\sum_{x'=0}^{(x-1)} I(x',y') = \sum_{x'=0}^{(x-1)} \sum_{y'=0}^{y} I(x',y') - \sum_{x'=0}^{(x-1)} \sum_{y'=0}^{(y-1)} I(x',y') \tag{13}$$

$$S(x,y-1) = \sum_{x'=0}^{x} \sum_{y'=0}^{(y-1)} I(x',y') \tag{14}$$

$$S(x-1,y) = \sum_{x'=0}^{(x-1)} \sum_{y'=0}^{y} I(x',y') \tag{15}$$

$$S(x-1,y-1)=\sum_{x'=0}^{(x-1)}\sum_{y'=0}^{(y-1)} I(x',y') \tag{16}$$

$$S(x,y)=S(x,y-1)+S(x-1,y)-S(x-1,y-1)+I(x,y) \tag{17}$$

Once this image has been compiled any rectangular sum, *RS*, can be computed by Equations (18) and (19) [53].

$$\begin{aligned} RS(x_o,y_o,x_l,y_l) &= S(x_o-1,y_o-1)+S(x_o+x_l-1,y_o+y_l-1)- \\ &\quad S(x_o-1,y+y_l-1)-S(x_o+x_l-1,y_l-1) \end{aligned} \tag{18}$$

$$\begin{aligned} RS(x_o,y_o,x_l,y_l) &= \sum_{x'=0}^{(x_o-1)}\sum_{y'=0}^{(y_o-1)} I(x',y')+\sum_{x'=0}^{(x_o+x_l-1)}\sum_{y'=0}^{(y_o+y_l-1)} I(x',y')- \\ &\quad \sum_{x'=0}^{(x_o-1)}\sum_{y'=0}^{(y_o+y_l-1)} I(x',y')-\sum_{x'=0}^{(x_o+x_l-1)}\sum_{y'=0}^{(y_o-1)} I(x',y') \end{aligned} \tag{19}$$

This method can be proven by separating each summation into multiple parts as shown below.

$$\begin{aligned} \sum_{x'=0}^{(x_o+x_l-1)}\sum_{y'=0}^{(y_o+y_l-1)} I(x',y') &= \sum_{x'=0}^{(x_o-1)}\sum_{y'=0}^{(y_o-1)} I(x',y')+\sum_{x'=x_o}^{(x_o+x_l-1)}\sum_{y'=0}^{(y_o-1)} I(x',y')+ \\ &\quad \sum_{x'=0}^{(x_o-1)}\sum_{y'=y_o}^{(y_o+y_l-1)} I(x',y')+\sum_{x'=x_o}^{(x_o+x_l-1)}\sum_{y'=y_o}^{(y_o+y_l-1)} I(x',y') \end{aligned} \tag{20}$$

$$\sum_{x'=0}^{(x_o-1)}\sum_{y'=0}^{(y_o+y_l-1)} I(x',y')=\sum_{x'=0}^{(x_o-1)}\sum_{y'=0}^{(y_o-1)} I(x',y')+\sum_{x'=0}^{(x_o-1)}\sum_{y'=y_o}^{(y_o+y_l-1)} I(x',y') \tag{21}$$

$$\sum_{x'=0}^{(x_o+x_l-1)}\sum_{y'=0}^{(y_o-1)} I(x',y')=\sum_{x'=0}^{(x_o-1)}\sum_{y'=0}^{(y_o-1)} I(x',y')+\sum_{x'=x_o}^{(x_o+x_l-1)}\sum_{y'=0}^{(y_o-1)} I(x',y') \tag{22}$$

By substituting these values back into Equation (19) and arranging summations all regions with the exception of the desired rectangle are eliminated.

$$\begin{aligned} RS(x_o,y_o,x_l,y_l) &= (1+1-1-1)\sum_{x'=0}^{(x_o-1)}\sum_{y'=0}^{(y_o-1)} I(x',y') + \\ &\quad (1-1)\sum_{x'=x_o}^{(x_o+x_l-1)}\sum_{y'=0}^{(y_o-1)} I(x',y') + \\ &\quad (1-1)\sum_{x'=0}^{(x_o-1)}\sum_{y'=y_o}^{(y_o+y_l-1)} I(x',y') + \\ &\quad \sum_{x'=x_o}^{(x_o+x_l-1)}\sum_{y'=y_o}^{(y_o+y_l-1)} I(x',y') \end{aligned} \tag{23}$$

$$RS(x_o,y_o,x_l,y_l) = \sum_{x'=x_o}^{(x_o+x_l-1)}\sum_{y'=y_o}^{(y_o+y_l-1)} I(x',y') \tag{24}$$

Using this equation it is possible to simplify the calculation of each feature to a weighted sum of its corner points.

The derivations of the vertical edge feature are shown below as an example of the simplification method.

$$h(1,x_o,y_o,x_l,y_l) = RS\left(x_o,y_o,\frac{x_l}{2},y_l\right) - RS\left(x_o+\frac{x_l}{2},y_o,\frac{x_l}{2},y_l\right) \tag{25}$$

$$\begin{aligned} h(1,x_o,y_o,x_l,y_l) &= \left[S(x_o,y_o) + S\left(x_o+\frac{x_l}{2},y_o+y_l\right) + \right. \\ &\quad \left. (-1)S\left(x_o+\frac{x_l}{2},y_o\right) + (-1)S(x_o,y_o+y_l)\right] - \\ &\quad \left[S\left(x_o+\frac{x_l}{2},y_o\right) + S\left(x_o+\frac{x_l}{2},y_o+y_l\right) + \right. \\ &\quad \left. (-1)S\left(x_o+\frac{x_l}{2}+\frac{x_l}{2},y_o\right) + (-1)S\left(x_o+\frac{x_l}{2},y_o+y_l\right)\right] \end{aligned} \tag{26}$$

$$h(1,x_o,y_o,x_l,y_l) = (1)\times S(x_o,y_o)+(-1-1)\times S\left(x_o+\frac{x_l}{2},y_o\right)+ (1)\times S(x_o+x_l,y_o)+(-1)\times S(x_o,y_o+y_l)+ (1+1)\times S\left(x_o+\frac{x_l}{2},y_o+y_l\right)+(1)\times S\left(x_o+\frac{x_l}{2},y_o+y_l\right) \tag{27}$$

$$h(1,x_o,y_o,x_l,y_l) = \left[S(x_o,y_o)+S(x_o+x_l,y_o)+2\times S\left(x_o+\frac{x_l}{2},y_o+y_l\right)\right]- \left[S(x_o,y_o+y_l)+S\left(x_o+\frac{x_l}{2},y_o+y_l\right)+2\times S\left(x_o+\frac{x_l}{2},y_o\right)\right] \tag{28}$$

A similar computation is developed for each type of feature. The resulting simplified equations are shown TABLE I, TABLE II, and TABLE III.

TABLE I Haar-like edge feature value calculation.

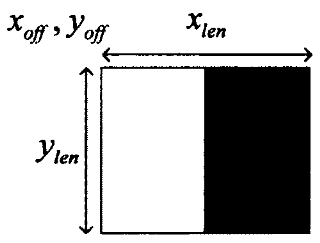
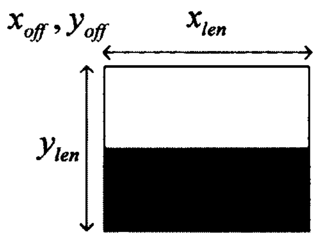
Feature	Diagram	Simplified Value
$h\begin{pmatrix}1,\\ x_{off},\\ y_{off},\\ x_{len},\\ y_{len}\end{pmatrix}$		$\begin{bmatrix} S\left(x_{off}, y_{off}\right)+ \\ S\left(x_{off}+x_{len}, y_{off}\right)+ \\ 2\times S\left(x_{off}+\frac{x_{len}}{2}, y_{off}+y_{len}\right)\end{bmatrix} - \begin{bmatrix} S\left(x_{off}, y_{off}+y_{len}\right)+ \\ S\left(x_{off}+x_{len}, y_{off}+y_{len}\right)+ \\ 2\times S\left(x_{off}+\frac{x_{len}}{2}, y_{off}\right)\end{bmatrix}$
$h\begin{pmatrix}2,\\ x_{off},\\ y_{off},\\ x_{len},\\ y_{len}\end{pmatrix}$		$\begin{bmatrix} S\left(x_{off}, y_{off}\right)+ \\ S\left(x_{off}, y_{off}+y_{len}\right)+ \\ 2\times S\left(x_{off}+x_{len}, y_{off}+\frac{y_{len}}{2}\right)\end{bmatrix} - \begin{bmatrix} S\left(x_{off}+x_{len}, y_{off}\right)+ \\ S\left(x_{off}+x_{len}, y_{off}+y_{len}\right)+ \\ 2\times S\left(x_{off}, y_{off}+\frac{y_{len}}{2}\right)\end{bmatrix}$

TABLE II Haar-like line feature value calculation.

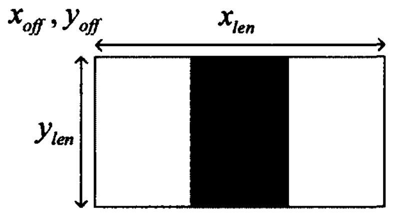
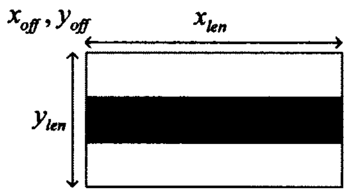
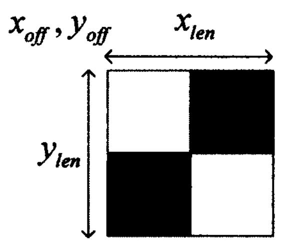
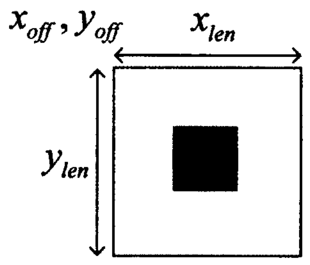
Feature	Diagram	Simplified Value
$h\begin{pmatrix}3, \\ x_{off}, \\ y_{off}, \\ x_{len}, \\ y_{len}\end{pmatrix}$		$\begin{bmatrix} S\left(x_{off}, y_{off}\right)+ \\ S\left(x_{off}+x_{len}, y_{off}+y_{len}\right)+ \\ 2\times S\left(x_{off}+\frac{x_{len}}{3}, y_{off}+y_{len}\right)+ \\ 2\times S\left(x_{off}+\frac{2}{3}x_{len}, y_{off}\right) \end{bmatrix} - \begin{bmatrix} S\left(x_{off}, y_{off}+y_{len}\right)+ \\ S\left(x_{off}+x_{len}, y_{off}\right)+ \\ 2\times S\left(x_{off}+\frac{x_{len}}{3}, y_{off}\right)+ \\ 2\times S\left(x_{off}+\frac{2}{3}x_{len}, y_{off}+y_{len}\right) \end{bmatrix}$
$h\begin{pmatrix}4, \\ x_{off}, \\ y_{off}, \\ x_{len}, \\ y_{len}\end{pmatrix}$		$\begin{bmatrix} S\left(x_{off}, y_{off}\right)+ \\ S\left(x_{off}+x_{len}, y_{off}+y_{len}\right)+ \\ 2\times S\left(x_{off}+x_{len}, y_{off}+\frac{y_{len}}{3}\right)+ \\ 2\times S\left(x_{off}, y_{off}+\frac{2}{3}y_{len}\right) \end{bmatrix} - \begin{bmatrix} S\left(x_{off}, y_{off}+y_{len}\right)+ \\ S\left(x_{off}+x_{len}, y_{off}\right)+ \\ 2\times S\left(x_{off}, y_{off}+\frac{x_{len}}{3}\right)+ \\ 2\times S\left(x_{off}+x_{len}, y_{off}+\frac{2}{3}y_{len}\right) \end{bmatrix}$

TABLE III Haar-like other feature value calculation.

Feature	Diagram	Simplified Value
$h\begin{pmatrix}5,\\ x_{off},\\ y_{off},\\ x_{len},\\ y_{len}\end{pmatrix}$		$\begin{bmatrix} S(x_{off},y_{off})+\\ S(x_{off},y_{off}+y_{len})+\\ S(x_{off}+x_{len},y_{off})+\\ S(x_{off}+x_{len},y_{off}+y_{len})+\\ 4\times S\left(x_{off}+\frac{x_{len}}{2},y_{off}+\frac{y_{len}}{2}\right)\end{bmatrix} - 2\begin{bmatrix} S\left(x_{off},y_{off}+\frac{y_{len}}{2}\right)+\\ S\left(x_{off}+\frac{x_{len}}{2},y_{off}\right)+\\ S\left(x_{off}+\frac{x_{len}}{2},y_{off}+x_{len}\right)+\\ S\left(x_{off}+x_{len},y_{off}+\frac{y_{len}}{2}\right)\end{bmatrix}$
$h\begin{pmatrix}6,\\ x_{off},\\ y_{off},\\ x_{len},\\ y_{len}\end{pmatrix}$		$\begin{bmatrix} S(x_{off},y_{off})+\\ S(x_{off}+x_{len},y_{off}+y_{len})+\\ 2\times S\left(x_{off}+\frac{2}{3}x_{len},y_{off}+\frac{y_{len}}{3}\right)+\\ 2\times S\left(x_{off}+\frac{x_{len}}{3},y_{off}+\frac{2}{3}y_{len}\right)\end{bmatrix} - \begin{bmatrix} S(x_{off},y_{off}+y_{len})+\\ S(x_{off}+x_{len},y_{off})+\\ 2\times S\left(x_{off}+\frac{x_{len}}{3},y_{off}+\frac{y_{len}}{3}\right)+\\ 2\times S\left(x_{off}+\frac{2}{3}x_{len},y_{off}+\frac{2}{3}y_{len}\right)\end{bmatrix}$

B. Statistically Boosted Classifiers

The AdaBoost was the initial classifier implemented by Viola and Jones [50] and is in many ways a much simpler approach to developing an object detector. If this threshold is met then a value proportional to the training error is produced.

Viola and Jones present a method with which both the optimal threshold and error can be computed in a single pass over a training set containing both positive and negative images. Each element of the training set is comprised of three items: the training image (*X*), the classification (*Y*), and a weight (*W*). The weight and classification are initialized as:

$$Y = \begin{cases} 0 & X \in \mathbf{N} \\ 1 & X \in \mathbf{P} \end{cases}, W = \begin{cases} \frac{1}{2M} & X \in \mathbf{N} \\ \frac{1}{2L} & X \in \mathbf{P} \end{cases} \tag{29}$$

M and *L* are the sizes of the negative training image set, $\mathbf{N}_,$ and positive training image set, $\mathbf{P}$, respectively. A weak classifier is added to the overall classifier in each iteration. To speed up the training process a subset of the feature set is initially chosen at random from the overall set.

The optimal threshold for each classifier is determined by the following algorithm. The response of each training image to the Haar-like feature is computed. A triple containing the value, classification, and weight is inserted into a list at the position as sorted by its feature value. This process is repeated for all training images resulting in a sorted response list. The total positive weight and the total negative weight are maintained as each value is calculated. The sorted list is then iterated through. The sum of positive weights and the sum of negative weights up to the particular position in the sorted list is computed. The errors for values greater than the threshold and less than the threshold are defined as:

$$\begin{aligned} ErrGT &= SumP + (TotN - SumN) \\ ErrLT &= SumN + (TotP - SumP) \end{aligned} \tag{30}$$

The threshold which generates the lowest error is select as the optimal threshold. The error associated with the optimal threshold is used to compute the value assigned to the individual classifier.

$$\alpha = \log \frac{1 - Err}{Err} \tag{31}$$

The classifier out of the reduced set with the lowest error is said to be the winner of the first round. A secondary set of similar features is computed and tested over the same training sets. The best feature for the secondary set is ultimately added to the classifier. The weight values for each training image are then updated to reflect the addition of the new weak classifier.

$$W(t+1) = W(t) \times \frac{Err}{1 - Err} \tag{32}$$

The process is then repeated until the desired number of features has been reached forming a monolithic AdaBoost classifier.

The main limitation to this technique is that a sub-frame of the image is either classified as a face or a non-face with no way of accurately determining the degree of confidence of the decision. Wu, et al. [54] attempt to extend the concept through the use of a "Real AdaBoosting" algorithm and a nested cascade structure. In their approach the response of each potential feature to a training set is normalized and classified into a discrete number of evenly spaced bins. Following training the probability of the feature responses in each bin correlating to a positive recognition is calculated. As a result the final classification result maps to a number in the range of $[-1,+1]$. The sign of this result still corresponds to a face if positive and a non-face if negative. However, the magnitude of the result is now correlated with the confidence in the decision. The limitation of this approach is the impracticality of the normalization of a large range of possible Haar-like feature responses. Also, the number of bins and the distribution of the Haar-like feature responses both factor into the accuracy of the sampling.

Fasel, et al. [52] present a generative framework for the real-time detection and classification of objects. In their approach a GentleBoost boosting algorithm is used to learn the likelihood ratios of an arbitrary object patch being of the object class or part of the background image. The example given in this publication first detects a face from an arbitrary frame and then subsequently works towards the location of the eyes within the facial sub-frame of the image. By searching for a specific context within a general context a higher level of precision can be achieved as the context models become smaller and smaller with each iteration.

This section seeks to combine the nested cascade structure from [54] with features learned from the GentleBoost algorithm presented in [52] to create a classifier for each region and landmark point in the overall feature tracker. The remainder of this section will outline the procedure for training each detector.

A training set, $\{(y_i,z_i):i=1,\ldots,M\}$, where y_i is a 24×24 pixel training image from either a positive example set, P, or a negative example set, N, is required to evaluate the classification performance of each feature in the features set. The category label z_i is assigned to each training image as follows:

$$z_i = \begin{cases} +1 & y_i \in P \\ -1 & y_i \in N \end{cases} \tag{33}$$

A large set of features, $H = \{h_k : k = 1,\dots,K\}$, forms the pool from which each classifier will be constructed. The notation $h_k(y_i)$ corresponds to the *kth* Haar-like feature in set H as evaluated on the integral image from image patch y_i according to the feature characteristics as defined by TABLE I, TABLE II, and TABLE III. Lastly, a weight, $D_t(i)$, is assigned to each example at the beginning of each iteration, t. The initial weight distribution assumes all training examples are weighted equally.

$$D_0(i) = \frac{1}{M} : i = 1,\dots,M \tag{34}$$

At each iteration, the feature which best improves the performance of the current classifier over the training set must be selected. For each feature in H a tuning function, $\eta : \Re \mapsto [-1,+1]$, is estimated which maps the feature response to the expectation of a correct response while taking into account the information from the existing classifier.

$$\eta\left(h_k\left(y_i\right)\right) = E^{D(i)}\left[Z \mid h_k\left(y_i\right)\right] : Z \in \{-1,+1\}, i = 1,\dots,M \tag{35}$$

The Nadaraya-Watson kernel regression method for density estimation is used to approximate the tuning function, η. Training examples are reduced to a set of triplets $\{(h_k(y_i), z_i, D_t(i))\} : i = 1,\dots,M$ in which $h_k(y_i)$ is the regressor variable, z_i is the label to be predicted, and $D_t(i)$ is the weight corresponding to the example. The resulting approximation is given by

$$\eta_k(\upsilon) = \frac{\sum_{i=1}^{M} z_i D_t(i) \kappa\left(\upsilon - h_k(y_i)\right)}{\sum_{j=1}^{M} D_t(j) \kappa\left(\upsilon - h_k(y_j)\right)} \tag{36}$$

such that $\kappa(.)$ is a Gaussian kernel centered over each feature response throughout the training set [55].

The feature and tuning function that have the lowest error over the training set must be found and added to the classifier. The overall error is found by

$$\rho = \sum_i \frac{t_i - p(y_i)}{\sqrt{p(y_i)\left(1 - p(y_i)\right)}} : t_i = \frac{z_i + 1}{2} \in \{0,1\}, \tag{37}$$

where $p(y)$ is the probability that image patch y belongs to its correct category and is defined by

$$p(y) = \frac{1}{1 + e^{\left\{-2\sum_j f_j(y)\right\}}}. \tag{38}$$

Each prospective feature is tested with the existing classifier. The feature which results in the lowest overall error is selected as the feature to be added for the current iteration.

$$\hat{k} = \arg\min_k \rho : f_t(y) = \eta_{\hat{k}}\left(h_k(y)\right) \tag{39}$$

$$f_t(y) = \eta_{\hat{k}}\left(h_{\hat{k}}(y)\right) \tag{40}$$

Once the new feature has been selected the training data distribution must be updated to reflect the change in the current classifier. The weights for each sample are lowered for each correctly classified sub-frame and are increased for incorrectly classified images. Z_t is a normalization factor which ensures the sum of the training data weights remains one.

$$D_{t+1}(i) = D_t(i)\frac{e^{-f(y)z}}{Z_t} : Z_t = \sum_i D_t(i)e^{-f(y)z} \tag{41}$$

The probability model can then be updated to include the additional feature:

$$p(y) = \frac{1}{1+e^{-2\sum_{n=1}^{t} f_n(y)}} \tag{42}$$

This process is repeated with each additional feature until the desired detection and false positive rates are achieved. The weight distribution of the training data is adjusted with the addition of each feature, pushing the emphasis toward those images which are incorrectly classified. Performance increases as features are added to the classifier.

Image patches which are similar to the target object receive a value of one since the majority of the classifiers $f_n(y)$ should produce positive values leading to a negative exponent. Conversely patches for which $f_n(y)$ result in negative values result in a positive exponent and a low probability.

TABLE IV Classifier responses and probability.

Classifier Response	$p(y)$
$-2\sum_{n=1}^{t} f_n(y) << 0$	1
$-2\sum_{n=1}^{t} f_n(y) = 0$	0.5
$-2\sum_{n=1}^{t} f_n(y) >> 0$	0

The probability metric can be remapped to the range of $[-1,1]$ through a linear manipulation as follows:

$$L(y) = 2p(y) - 1 = \frac{1 - e^{-2\sum_{n=1}^{t} f_n(y)}}{1 + e^{-2\sum_{n=1}^{t} f_n(y)}} \tag{43}$$

Evaluation of a prospective sub-frame using the above equation will result in the classification of an object if $L(y) > 0$ or a non-object if $L(y) \leq 0$, both with a certainty of $|L(y| \in [0,1]$.

C. *Cascade Training*

Viola and Jones [50] first presented the idea of converting a monolithic classifier into a cascade structure. The overall detection rate, D^{*}_{tot}, and false positive rate, F^{*}_{tot}, of a cascade of classifiers can be computed as $D^{*}_{tot} = \prod_{l} D^{*}$ and $F^{*}_{tot} = \prod_{l} F^{*}$, where D^{*} and F^{*} are the minimum detection and maximum allowable false positive rates per level l. To achieve a high overall detection rate each stage must maintain a high detection rate. However, to maintain a low overall false positive rate each stage needs only a modest false positive rate. The following example illustrates the advantage of using a cascade to achieve an overall detection rate of .9 and an overall false positive rate of on the order of 10^{-6} using a ten-layer cascade.

$$D^{*}_{tot} = \prod_{l=1}^{10} .99 = (.99)^{10} = .9 \quad (44)$$

$$F^{*}_{tot} = \prod_{l=1}^{10} .30 = (.30)^{10} = 6\times10^{-6} \quad (45)$$

Each layer must have only a 30% false positive rate. It is then possible to begin with a weaker classifier and work up to stronger classifiers which require more Haar-like features. As a result only sub-frames which closely resemble the target image must be evaluated by the majority of the features in the cascade.

The training process for developing a detection cascade involves adding monolithic classifiers until a target false positive rate is met, the maximum number of layers is reached, or there are no longer any negative training images remaining in the database. The first stage in the training process involves determining how many weak classifiers should be added to the current layer at a time. Early on this number is held relatively low. However, due to the increased difficulty of training the later layer, it becomes necessary to add more weak classifiers to the individual layer at a time in order to speed up the training time. In this particular implementation the first seven-layer work from two classifiers per addition rounds up to 50 classifiers after the seventh layer.

Each layer is built up until the current layer's false positive rate is met or the maximum number of overall classifiers has been reached. The designated number of weak classifiers is added to the current layer. The current layer is then tested on the evaluation set to determine the current detection rate. If the detection rate is not met the threshold for a positive frame is reduced by 5%. This process is repeated until the detection rate for the layer is met or the threshold becomes zero. Next the current layer with the possibly reduced threshold is tested against the negative training set to determine the current false positive rate for the layer. During this evaluation true negatives are noted through the use of the flag contained in the *TrainIm* structure. If the current false positive rate for the layer is above the acceptable rate for the layer then another round of features is added and the above process is repeated. When the current layer is established it is added to the detection cascade.

At this point the negative training set must be updated to reflect the new cascade. To do this each true negative flagged previously is eliminated from the training set. Next, new images are randomly selected and evaluated using the entire detection cascade. Any image which makes it through the entire cascade is added to the negative dataset. This ensures that at any time the negative training set contains only false positive images. If an image is rejected the prospective training image is thrown away and a count of overall false positives is updated. Once the set has been rebuilt to its previous size the overall current false positive rate is calculated. If this rate meets the target rate then training is complete. Otherwise another layer must be added and the entire process is repeated. Fig. 6 outlines the algorithm described in this section. The syntax *L_FPR* = *L(N)* implies the false positive rate as calculated by applying the negative training set to the individual layer.

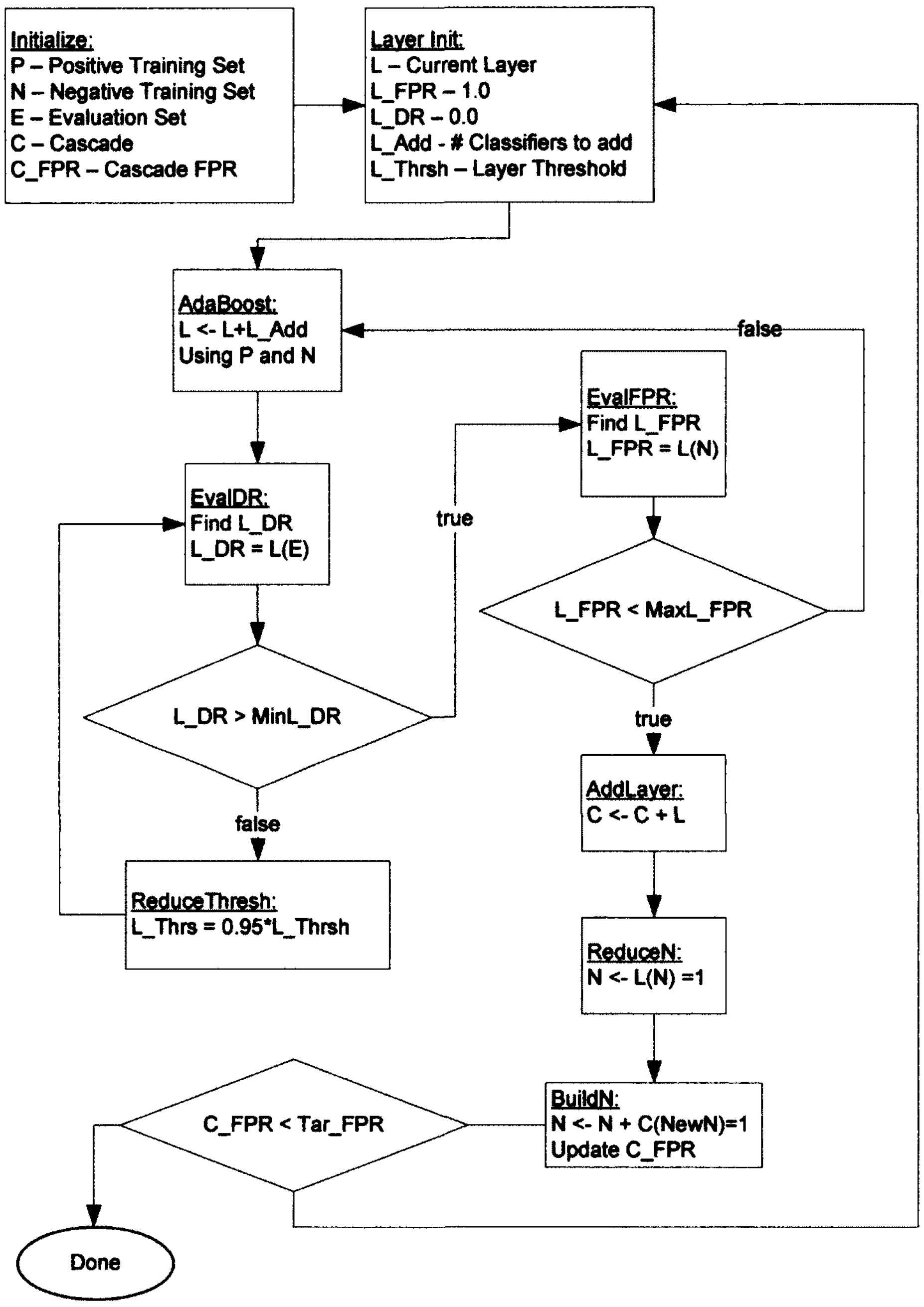


Fig. 6 Cascade training algorithm.

D. Cascaded Classifiers

Wu, et al. [54] extend this concept one step further. They realized that each individual layer had no knowledge of the work of the previous layers beyond a reduction of false positives in the training sets. To overcome this deficiency they manipulate the cascade structure. The first weak classifier of each layer in the overall cascade is created from the previous layer. Fig. 7 shows the structure of a nested cascade. Weak classifiers based upon the previous layer are shown as hexagons while standard Haar-like feature classifiers are shown as circles. In this manner information learned from previous layers of the cascade is not lost and does not need to be reincorporated in the current layer. The resulting nested cascaded detector contains fewer layers and fewer Haar-like features equating to a higher level of efficiency.

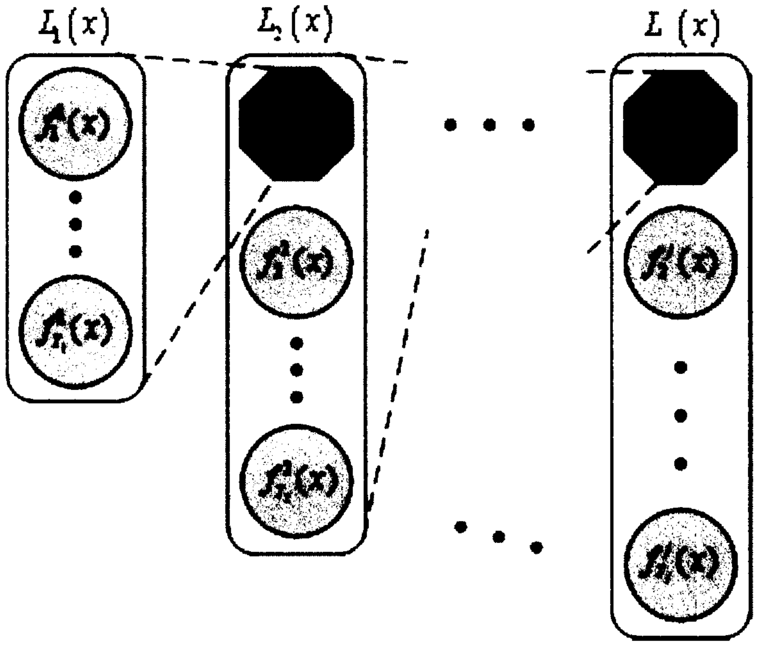


Fig. 7 Nested cascade structure.

To train the nested cascade classifier, the first desired maximum false positive rate per layer, $F_{\max}^{*}$, the minimum detection rate per layer, $D_{\min}^{*}$, the targeted false positive rate, F_{tar}^{*}, and positive and negative training sets, P and N, must be defined. The entire cascade is noted as $C(x)$ and contains j layers, $L_j(x)$, as shown in Fig. 7. The cascade detection and false positive rates, D_{tot}^{*} and F_{tot}^{*}, are initialized at one to be updated with the addition of each feature.

The first layer is trained using the GentleBoosting technique described above until the conditions specified by $D_{\min}^{*}$ and $F_{\max}^{*}$ are met. The actual detection and false positive rates, D_{act}^{*} and F_{act}^{*}, are evaluated over the P and N training sets. The first layer is added to the cascade and the performance rates are then updated:

$$D_{tot}^{*} = D_{tot}^{*} \times D_{act}^{*} \tag{46}$$

$$F_{tot}^{*} = F_{tot}^{*} \times F_{act}^{*} \tag{47}$$

The N training set is then reduced by eliminating any negative training sample that is correctly classified by the current cascade. In this way attention is focused towards the more difficult examples as training progresses.

Remaining layers of the cascade have an additional first step. The L_{i-1} layer is used as the first weak classifier of layer L_i and the sample distribution $D_t(i)$ is updated accordingly. The remainder of the layer is trained in the same manner as the first. Layers are added until the overall false positive rate falls below the target false positive rate, $F_{tot}^{*} < F_{tar}^{*}$.

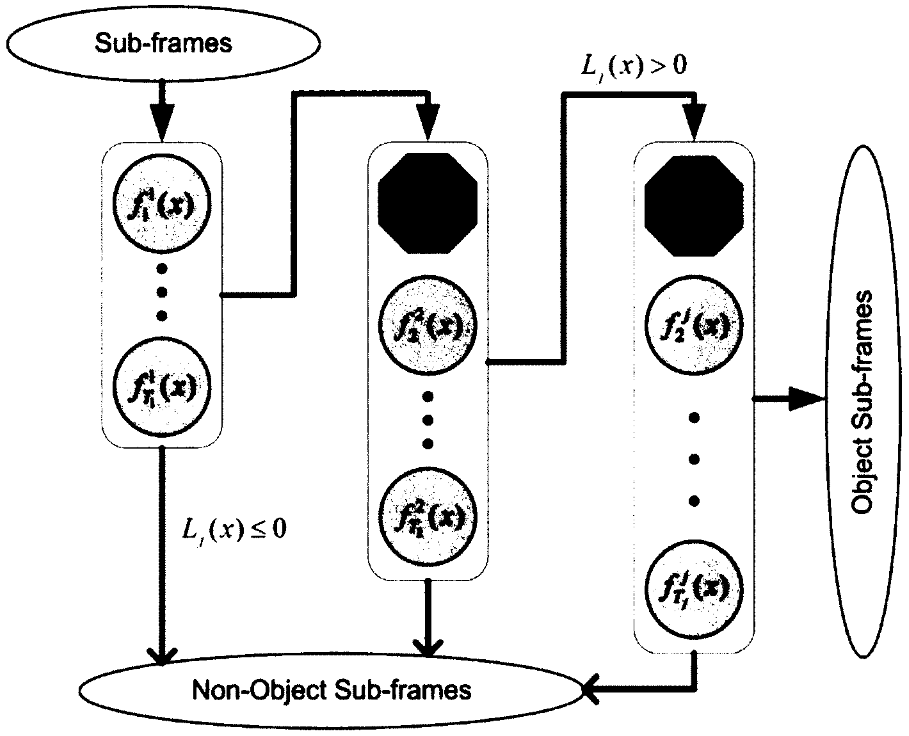


Fig. 8 Nested cascade operation.

Each prospective sub-frame enters the first layer of the classifier. If the score from the current layer is greater than one then the sub-frame proceeds to the next layer, which is a stronger classifier consisting of more features. If at any point the score drops below zero then that frame is immediately rejected as a non-object and no further processing occurs. If a sub-frame reaches the end of the cascade it is classified as an object with a confidence score, $conf_{C(x)} = \prod_j L_j(x)$, where $L_j(x)$ is the response of the layer to image section x.

$$L_1(x) = \sum_{t=1}^{T_1} f_t^1(x) \tag{48}$$

$$L_j(x) = L_{j-1}(x) + \sum_{t=1}^{T_j} f_t^j(x) \tag{49}$$

By calculating the classification confidence in this manner, the confidence score from previous detection layers is factored into the calculation of the final detection.

V. Sub-frame Reduction for Preprocessing in Object Detection

High-definition video is becoming more available for a wide variety of applications as memory becomes cheaper and sensor arrays smaller. Among these possible applications is that of facial expression recognition. More detail can be extracted from high-definition frames and images, ideally leading to higher recognition rates. Face recognition can be divided into three separate stages: detection of face regions from the frame or image, facial feature extraction and classification, and recognition and identification [56]. The main tradeoff of using the larger, higher-definition frames is the increase in possible sub-frames which may contain a face. As a result, the processing time for a single frame is increased drastically.

This chapter presents a preprocessing approach to eliminate low-variance regions of a frame to reduce the overall area which must be searched. A quad-tree approach is taken to perform this elimination as quickly as possible. A quad-tree is a method of dividing an image into smaller sections by repeatedly quartering the image. Quad-tree data structures are an established technique for coding and compressing images and video into separate spatial regions [57]. Saupe and Jacob present a variance-based quad-tree structure for use in fractal image compression [58]. The ability of quad-trees to quickly and efficiently group together similar areas has been exploited in several object detection algorithms. Barroso, et al. use quad-trees by grouping together regions of similar color to locate and identify vehicle license plates [59]. Kaplan, et al. use the multi-resolution properties of a quad-tree image to detect broadside targets during image formation on ultra-wide-band synthetic aperture radar images [60]. Park and Park use a quad-tree structure to efficiently store templates which are easily resizable for comparison to an input image, detecting possible faces using template-based face detection [56].

The goal of this chapter is to present a preliminary algorithm for the elimination of unnecessary sub-frames while using Viola-Jones face detection. In its full implementation the face detection algorithm searches many scales and offsets of an integral image. A specific sub-frame is determined to be a face if it passes through a cascade of increasingly stronger classifiers without being eliminated. Each classifier is constructed using a set of statistically determined Haar-like features [50]. Applying each integral image sub-frame to the detection cascade costs time and resources, especially as the number of sub-frames increases for higher-definition frames. The algorithm presented will first eliminate all areas of the frame with a regional variance below a predetermined threshold. The result of this step is the creation of a mask specifying only valid search regions. The mask is filled using an averaging filter followed by binary thresholding. Finally, an integral image is formed using the filled mask. This integral mask is used to determine if a sub-frame contains a minimal percentage of valid area to be considered for the Viola-Jones detector. If deemed valid, the integral image sub-frame from the original frame is passed into the face detector cascade.

The remainder of this chapter is organized as follows. Section A presents the sub-frame elimination algorithm. Preliminary experimental results and analysis are presented in Section B. Section C offers conclusions and future research work.

A. Regional Variance Sub-frame Reduction

This technique can be divided into four steps. First, regional variance is used to dynamically prune a quad-tree representation of the frame under examination. Second, as child branches are trimmed off, the area covered by that sub-frame is eliminated in a mask. Third, the mask is smoothed and filled. Finally, the filled mask is used to determine if a given sub-frame has enough remaining valid area to still be considered for face detection.

1) Regional Variance Calculation

Using a quad-tree data structure, the largest possible regions can be eliminated first. No further evaluations are required for that particular area of the frame. Thus, low-variance regions are eliminated largest down to smallest. First, the frame to be tested must be converted to a grayscale image of the same size. Next, an integral image is created to speed up the computation of regional variance of an arbitrary sub-frame and for processing by the Viola-Jones detection cascade. An integral image, *ii*, is formed by summing all pixel values above and to the left of the current location in the original grayscale frame *i* as specified by:

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x',y') \tag{50}$$

As a result, the sum of pixels in a generic sub-frame can be quickly determined through the addition and subtraction of corner values in the integral image. The following formula can be used to find the total sum of pixel values in the rectangle shown in Fig. 9:

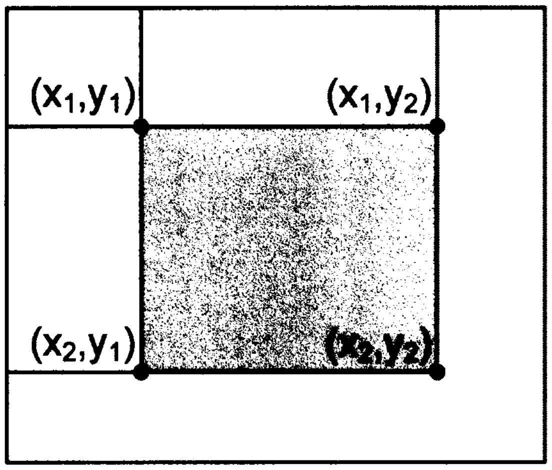


Fig. 9 Regional variance calculated using an integral image for an arbitrary sub-frame.

$$S(x_1,x_2,y_1,y_2) = ii(x_2,y_2) - ii(x_1,y_2) - ii(x_2,y_1) + ii(x_1,y_1) \tag{51}$$

With the sum of the pixels in the sub-frame known, the mean pixel value can be quickly computed by dividing the sum by the number of pixels in the sub-frame.

$$\mu(x_1,x_2,y_1,y_2) = \frac{S(x_1,x_2,y_1,y_2)}{(x_2-x_1)(y_2-y_1)} \tag{52}$$

The variance, σ, for the sub-frame can be computed using the mean value and the original sub-frame pixel values.

$$\sigma(x_1,x_2,y_1,y_2) = \frac{1}{(x_2-x_1)(y_2-y_1)} \sum_{x_1 \le x \le x_2} \sum_{y_1 \le y \le y_2} \left(i(x,y) - \mu\right)^2 \tag{53}$$

Finally, the regional variance, σ_r, for a sub-frame is calculated by normalizing the variance by the mean computed by Equation (52).

$$\sigma_r = \frac{\sigma}{\mu} \tag{54}$$

By performing this normalization the regional variance of the darker regions is increased at a greater rate than that of a lighter sub-frame with the same variance. Since the darker regions generally form features used in identifying faces, this parameter reduces the likelihood of the rejection of these sub-frames. Likewise, areas with a low variance and high mean (e.g., sky and walls) will yield a low regional variance value.

2) Quad-tree Formation

To quickly eliminate the maximal areas of the frame below a pre-specified threshold, a quad-tree data structure is used. The head node is set to the complete frame. The frame is then divided into four equally-sized sub-frames numbered with the top left, top right, bottom left, and bottom right segments taking positions one through four, respectively. This same procedure is then repeated iteratively for each of the four child nodes until a complete tree, l levels deep, is constructed. Note that each node can be addressed by an ordered set of integers of the set $\{1,2,3,4\}$. Fig. 10 shows an example of this concept.

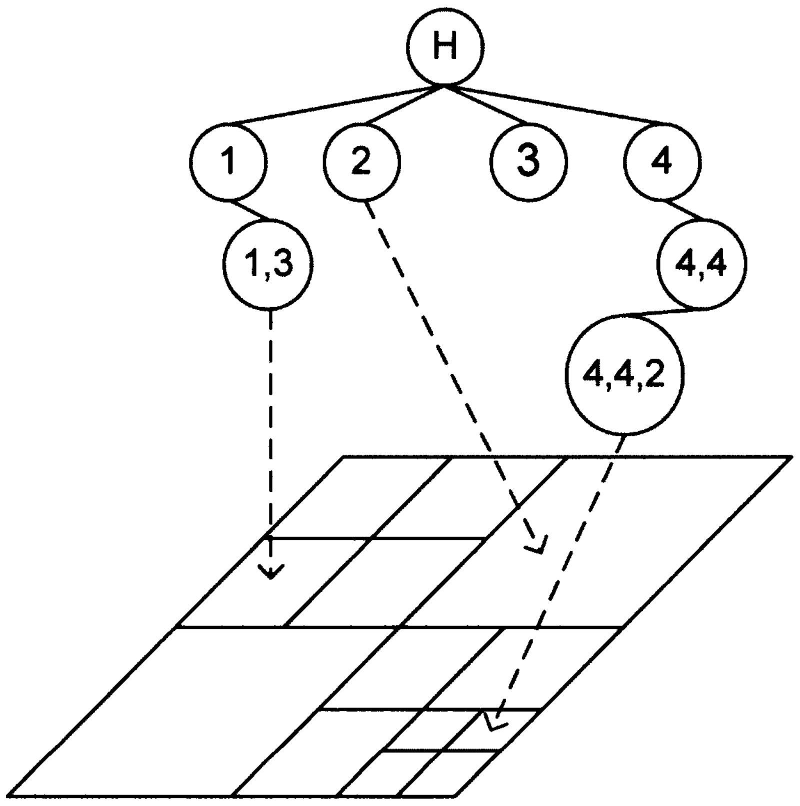


Fig. 10 Quad-tree deconstruction of a frame.

The procedure above creates a complete quad-tree down to a maximum of l levels. The tree can be pruned based on regional variances calculated for each sub-frame as referenced by each node in the quad-tree. However, this approach will result in many unnecessary computations. The proposal in this paper works on the presumption that once an area has a regional variance below the threshold, τ , all sub-regions of that area may also be eliminated. By building the tree incrementally as the regional variance is computed, only necessary computations will be performed.

Forming the tree incrementally requires considerable forethought as there are many cases and conditions which must be taken into account. A basic layout of the algorithm used to create the quad-tee structure is presented in Fig. 11. First the regional variance is computed for the entire frame by the method described in Section V.A.1). The quad-tree is then initialized with the whole image set to the head node. The current node in the system is also set to the head node.

The remainder of the quad-tree construction algorithm is accomplished in two steps. The current node of interest is divided into four equal sub-regions and the regional variance is computed for each sub-region. If the regional variance is above τ , then a new child node is added to the tree in the direction of that segment. If the regional variance is below τ , the region in consideration is set to all zeros in a mask image. The mask used here is the same size as the frame with all pixel values initialized to 255.

Several key pieces of information must be handled to correctly decide which sub-quadrant should be examined next. The next branch to be taken, the last branch of the current node, the current depth or level of the tree, and the position of the current node must all be maintained.

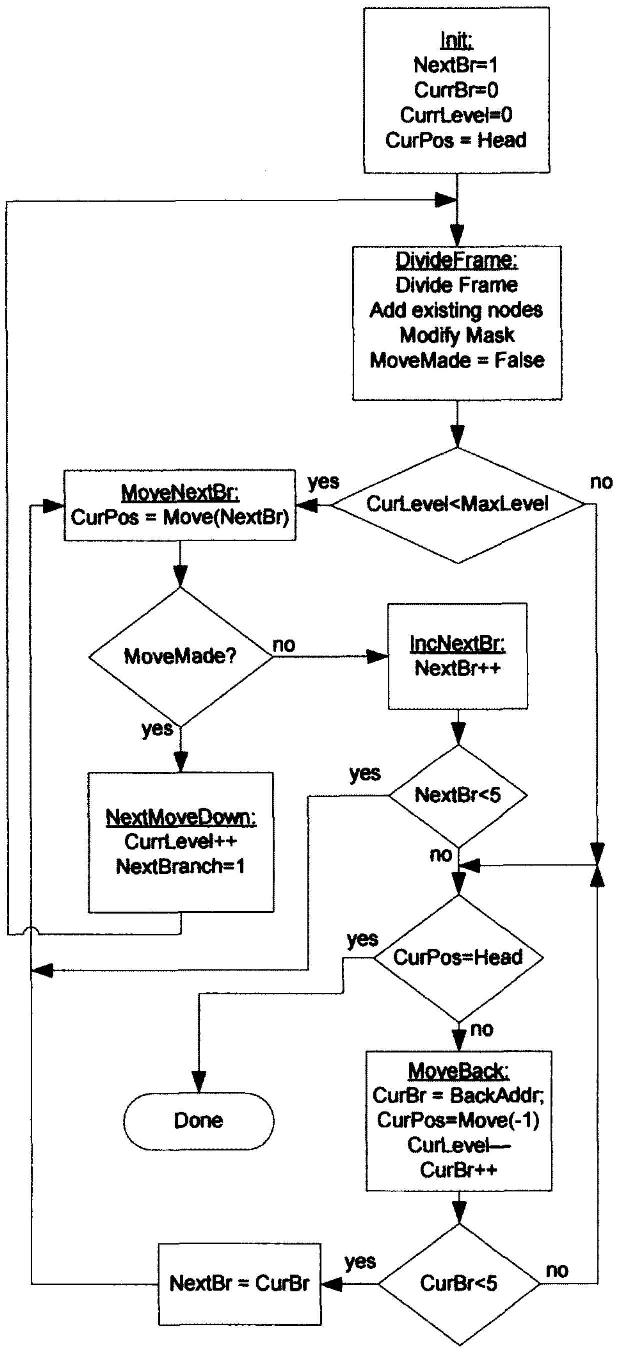


Fig. 11 Quad-tree building algorithm.

The tree always seeks to move the left and down as long as nodes exist in those directions and the current level is below a specified maximum level *l*. When the tree reaches maximal depth or has no more child nodes to break down, then it must backtrack, moving back up the tree. The "Move" function attempts to move the child node specified by the branch direction or the parent node if given a "-1" argument. If this child node exists, then the move is made, altering the current position. Otherwise, the next child node is checked. If no children remain, *NextBr* equals five, then the current position must backtrack. To move across a level, the current branch is noted and the current position is moved to the parent. The current branch, just noted, is then incremented and if it less than five, it becomes the next branch attempted. If this branch is equal to five then the current position is moved up one level. If the current positions is equal to the head node and the next branch is equal to five, then there are no more children to process and the tree is said to be completed as specified by the done stage.

The end result is a quad-tree containing all areas with a regional variance above that of the threshold, τ , and a mask with the region below this threshold removed. The resulting mask must then be further processed to smooth and fill smaller gaps of low regional variance.

3) Mask Smoothing and Filling

To make the mask, $M(x,y)$, more uniform and eliminate any small holes or leftover smaller "invalid" regions, the mask must be smoothed and filled. This process occurs in two phases. The mask is blurred using a square averaging filter of odd width and height, β . The resultant smoothed mask, $Ms(x,y)$, is specified by:

$$Ms(x,y)=\frac{1}{\beta^2}\sum_{x-\left\lfloor\frac{\beta}{2}\right\rfloor\le x'\le x+\left\lfloor\frac{\beta}{2}\right\rfloor}\;\sum_{y-\left\lfloor\frac{\beta}{2}\right\rfloor\le y'\le y+\left\lfloor\frac{\beta}{2}\right\rfloor} M(x',y') \qquad (55)$$

The smoothed mask, $Ms(x,y)$, is then subjected to a binary threshold, α, to create a filled mask, $Mf(x,y)$.

$$Mf(x,y)=\begin{cases}0 & Ms(x,y)\le\alpha\\ 1 & Ms(x,y)>\alpha\end{cases} \tag{56}$$

The parameters β and α together determine the minimum sizes of gaps that will be filled and the width of the border which will surround remaining higher-variance regions.

4) Face Detection

With the processed mask containing only valid regions computed, face detection can now be performed. First, an integral image of the filled mask, $MfII(x,y)$, must be created. Next, three parameters must be set to tune the speed and accuracy of the detector. The first parameter is that of the minimum ratio of a sub-frame that must be valid for that sub-frame to be considered for face detection, ρ_{min}. Since $Mf(x,y)$ can only contain values from the set $\{0,1\}$, the ratio of valid area to total area can be computed by using the Equation (51)-(52) with $MfII(x,y)$ as the base integral image. If this ratio is greater than that of ρ_{min}, then the sub-frame will be passed on to the face detection cascade. If less than ρ_{min}, the sub-frame will be discarded and not considered for face detection.

The number of total sub-frames that will be subjected to this check is related to the two internal Viola-Jones parameters. The first is the step size, δ, which determines the degree of offset between sub-frames in both the horizontal and vertical directions. The smaller this parameter is, the finer the search, at a tradeoff of longer computation time. The initial trained classifier is set to search only for a predefined size of face. It must be scaled to search for larger faces. The cascade size after up-scaling determines the size of the sub-frame which must be passed to the test frame. The second parameter, υ, defines the percentage for which the size of the search sub-frame is increased each time the scale is increased. This value must be greater than or equal to one. For example, an upscale value of 1.5, applied to a cascade with an initial size of 24×24, would yield a search window of size 36×36 for the second scale and 54×54 for the third scale, etc., until the search window size is greater than either the height or width of the test frame. Thus, the closer this value is to one, the more different face sizes are searched, with a tradeoff of a longer required search time.

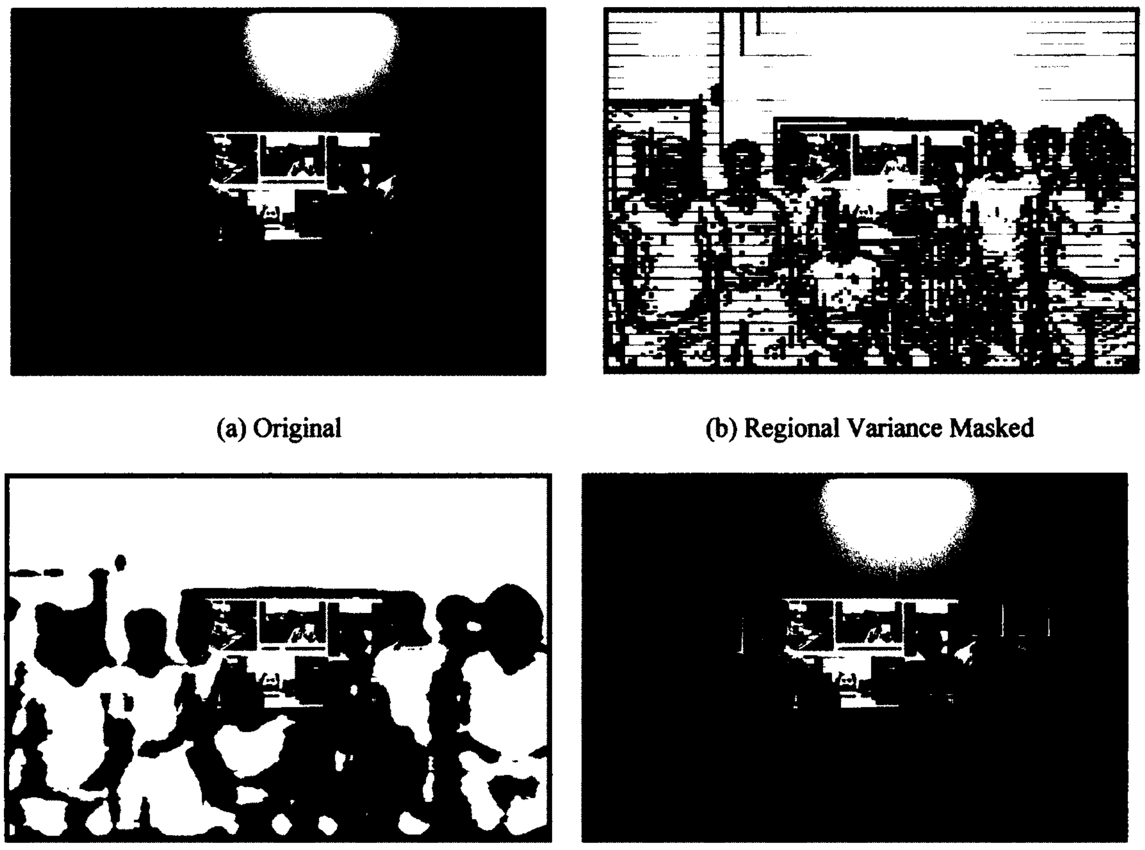
B. Preliminary Testing

The regional variance sub-frame reduction algorithm is implemented in C++, making use of the OpenCV Viola-Jones library. The OpenCV Viola-Jones frontal face detection cascade was utilized for detection purposes only if a sub-frame was deemed valid by the preprocessing stage. Currently, the system is set to operate on a single frame or image. The parameter settings for each individual image are determined empirically. Further testing is still required over a wider set of test images to determine optimal automatic settings for the parameters of the system. The final result and intermediate stages of an example image are shown in Fig. 12. The parameters of this test are given in TABLE V.

TABLE V Parameter values for example in Fig. 12.

Parameter	Value
τ	.1
L	6
β	21
α	150
ρ_{min}	.90
δ	5
υ	1.2

A deep tree depth is used to achieve a tight fit around high-variance regions. The use of a deep tree requires a low regional variance threshold so as to not eliminate facial regions. As a result, many sub-frame regions are not eliminated until they are on the lower levels of the tree. While this requires more computations, a tighter fit can be achieved and more overall area can be removed. With this tighter fit a higher minimum ratio, ρ_{min}, can also be used, further eliminating more sub-frames. A step size of five and a percent upscale value of 1.2 were used. For larger images these parameters could be altered to increase the speed of detection with the possible side effect of possible reduced accuracy.



(a) Original (b) Regional Variance Masked

(c) Filled Masked (d) Detection Result

Fig. 12 Example image processed with subframe reduction tecnique.

With the parameter settings presented in TABLE V, 22 face frames were detected, include all nine of the principle faces in the frame. Multiple recognitions could be grouped together in future implementations. 14,388 of the 104,238 total possible sub-frames were actually set to the cascade, equating to 86.2% of the sub-frames being eliminated from consideration for detection. This particular test image contains a large amount of low-variance space, resulting in a high percentage of sub-frame reduction. Scenes with more high-variance areas may not see as much sub-frame reduction.

TABLE VI Example image performance results.

Parameter	Values
Total Preprocessed Sub-frames	104,238
Total Remaining Sub-frames	14.388
Percent Sub-frame Reduction	86.2%
Face Detection Rate	100%

C. Summary

An efficient quad-tree approach for regional variance-based sub-frame reduction was introduced for the application of face detection in high-definition video frames. A quad-tree was first used to break down the frame, eliminating areas with a regional variance below that of a specified threshold created in an image mask. The mask was then smoothed and thresholded, cleaning up the mask image by applying neighborhood characteristics. Finally, the smoothed mask was used to determine if a sub-frame was valid and should be searched for a face. An example was presented showing a preliminary result suggesting the feasibility of this particular approach with proper parameter selection.

In frames which contain large areas of low variance area the potential exist for a reduction in computational time. The creation of the initial integral image is directly related to the original size of the frame and must be computed for the Viola-Jones face detection algorithm. Additional processing is required for each sub-frame of the quad-tree encoded frame to calculate the regional variance. This calculation is related to the size of each sub-frame. However, since the regional variance of large sub-frames from the top of the tree are calculated first and eliminated if their variance falls below a threshold large portions of the image can be removed from search contention. The final smoothing step is also dependant on the size of the frame and the averaging kernel being used. In the case were no sub-frames could be described as having a computational complexity of $O((L+2)N)$, where L is number of levels of the tree depth and N is the number of pixels in the sub-frame. However if sub-frames are removed due to low variance the computational complexity becomes, $O((L+2)N-R)$, where R is the combined regions eliminated due to low regional variance and the sub-frame that make up the child branches of those nodes.

Further work is required to test this technique over wider datasets to determine the optimal parameterization and how much detailed tuning would be required for different environments. Future research will work to find methods of automatically determining optimal values for all parameters based on the faces used for training the Viola-Jones cascade and input scene characteristics. Also, the current variance threshold must be set low if a high-depth tree is utilized. This condition results in elimination of many smaller areas and requires more computations to be performed. By using an independent regional threshold for each level of the quad-tree, larger portions of the frame may be eliminated, saving computations. More work is required determine in what situations this approach can be most effectively utilized. Analogous to the difficulty a human being would have in complex environments with a small or low-variance area, potential performance gains of the sub-frame reduction algorithm may be minimized.

VI. A Real-Time Emotion Detection By A Binary Decision Tree Approach

The computer has become more and more integrated into our lives. Yet, when it comes to the world of computers, there are situations where the man-machine interaction could be improved by having machines capable of adapting to their users. The term "human computer interaction" suggests a two-way exchange, with each participant aware of the other and responding appropriately. However, today's computers may appear frequently rude and indifferent, as they are almost completely unaware of the actual state of the human user.

Human-Computer Interaction (HCI) is a research area aiming at making the interaction with computer systems more effective, easier, safer, and more seamless for the users. The goal of this chapter is to contribute to the development of an HCI environment in which the computer detects and tracks the user's affective states and initiates communications based on this knowledge, rather than simply responding to user commands.

The remainder of this chapter is organized as follows. Section A gives background and related work to the devolvement of HCI interfaces. Section B discusses the design and function of the system developed. The results and performance of the system are presented in Section V.C. The final section presents a summary and future work.

A. Background

In the scientific community, a shared belief is that the next step in the advancement of computing devices and user interfaces is not to simply make faster applications but also to add more interactivity, responsiveness, and transparency to them. In the last decade much more effort has been directed towards building multi-modal, multi-media, multi-sensor user interfaces that emulate human-human communication with the overall long-term goal of transferring natural means and expressive models of communication to computer interfaces [7].

Cross-disciplinary approaches have begun to develop user-oriented interfaces that support non-GUI interaction by synergistically combining several simultaneous input and/or output modalities, referred to as multimodal user interfaces. In particular, multimodal PUIs have emerged as a potential candidate for the next interaction paradigm due to two key features. First, PUIs are highly interactive. Unlike traditional passive interfaces that wait for users to enter commands before taking any action, perceptual interfaces actively sense and perceive the world and take actions based on goals and knowledge at various levels. Ideally, this is an "active" interface that uses "passive," or non-intrusive, sensing. Second, they are multimodal, making use of multiple perceptual modalities (e.g., sight, hearing, touch) in both directions: from the computer to the user, and from the user to the computer. PUIs move beyond the limited modalities and channels available with a keyboard, mouse, and monitor to take advantage of a wider range of modalities, either sequentially or in parallel [8].

Facial expressions are the facial changes in response to a person's internal emotional states, intentions, or social communications. Facial expression analysis has been an active research topic for behavioral scientists since the work of Darwin in 1872 [1, 61-63]. Suwa, et al. presented an early attempt to automatically analyze facial expressions by tracking the motion of 20 identified spots on an image sequence in 1978 [64]. Much progress has been made to build computer systems to help us understand and use this natural form of human communication [65].

The process of reading the mind in the face is inherently uncertain. People can express the same mental state using different facial expressions. Moreover, the recognition of affective cues is in itself a noisy process. To account for this uncertainty, Kaliouby, et al. use a multi-level representation of the video input, combined in a Bayesian inference framework, specifically the dynamic Bayesian networks, for developing their mind-reading machine [34]. They used the Mind Reading DVD, a computer-based guide to emotions, developed by a team of psychologists led by Simon Baron-Cohen at the Autism Research Centre in the University of Cambridge, to train the statistical classifiers in their inference system [66]. This DVD was originally designed to help individuals diagnosed along the autism spectrum recognize facial expressions of emotions.

The DVD is based on the taxonomy of emotion by Baron-Cohen, et al. that covers a wide range of affective and cognitive mental states. The taxonomy lists 412 mental state concepts, each assigned to one (and only one) of 24 mental state classes [67]. Out of the 24 classes, they have focused on the automated recognition of six classes that are particularly relevant in an HCI context, and that are not in the basic emotion set. The six classes are: agreeing, concentrating, disagreeing, interested, thinking, and unsure. This project will focus on identifying the same six states with the addition of the basic emotional state of angry, utilizing an efficient binary decision tree.

Kaliouby's system abstracts raw video input into three levels. First, actions are explicitly coded based on the facial feature point return from the FaceTracker [32]. Displays are then recognized by HMMs. Mental states are assigned probabilities by dynamic Bayesian networks. The facial feature tracker Kaliouby used, FaceTracker, is part of Nevenvision's commercial facial feature tracking software development kit (SDK) [68]. FaceTracker uses a generic face template to bootstrap the tracking process, initially locating the position of 22 facial landmarks. To track the motion of the points over a live or recorded video stream, the tracker uses a combination of Gabor wavelet image transformations and NNs.

B. System Design

Facial expression analysis includes both measurement of facial motion and recognition of expression. The general approach to automatic facial expression analysis consists of three steps (see Fig. 13): face acquisition, facial feature extraction, and facial expression recognition. Face acquisition is a processing stage to automatically find the face region for the input images or sequences. It can be a detector to detect faces for each frame or just detect a face in the first frame and then track that face in the remainder of the video sequence.

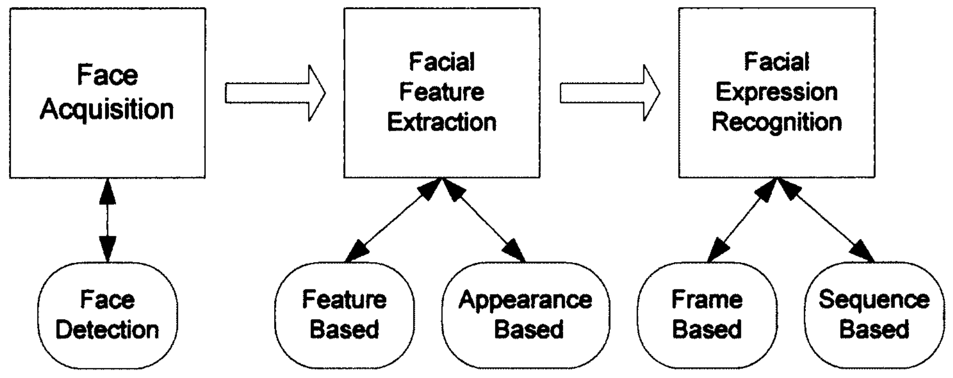


Fig. 13 Conceptual diagram.

After the face is located, the next step is to extract and represent the facial changes caused by facial expressions. In facial feature extraction for expression analysis, there are mainly two types of approaches: geometric feature-based methods and appearance-based methods. The geometric facial features present the shape and locations of facial components including but not limited to the mouth, eyes, brows, and nose. The facial components or facial feature points are extracted to form a feature vector that represents the face geometry. With appearance-based methods, image filters, such as Gabor wavelets, are applied to either the whole face or specific regions in a face image to extract a feature vector. Depending on the different facial feature extraction methods, the effects of in-plane head rotation and different scales of the faces can be eliminated by face normalization before the feature extraction or by feature representation before the step of expression recognition.

Facial expression recognition is the last stage of automatic facial expression analysis systems. The facial changes can be identified as facial AUs or prototypic emotional expressions. Depending on whether the temporal information is used, the facial expression can be classified as frame-based or sequence-based.

1) Face Acquisition

Object detection, in particular face detection, is an important element of various computer vision areas, such as image retrieval, video surveillance, HCI, etc. The goal is to find an object of a predefined class in a static image or video frame. Sometimes this task can be accomplished by extracting certain image features, such as edges, color regions, textures, contours, etc., and then using some heuristics to find configurations and/or combinations of those features specific to the object of interest. But for complex objects, such as human faces, it is hard to find features and heuristics that will handle the huge variety of instances of the object class (e.g.: faces may be slightly rotated in all three directions; some people wear glasses; some have moustaches or beards; often one half of the face is in the light and the other is shadow, etc.). For such objects, a statistical model (classifier) may be trained instead and then used to detect the objects. In the emotion detection system presented in this chapter, Viola-Jones face detection is used to extract facial regions. This technique is presented in detail in Chapter IV.

2) Facial Feature Extraction

A number of studies have shown that the visual properties of facial expressions can be described by the movement of points belonging to facial features [48, 69, 10]. These feature points are typically located on the eyes and eyebrows for the upper face, the lips and nose for the lower face. Fig. 14 illustrates the 2D face model of the 23 feature points used throughout this chapter. The position of the feature points is located in each frame by automatically matching a manually annotated image of the individual to each frame and creating an AAM as discussed in Chapter III.

To implement the AAM to perform tracking, a training set consisting of 1066 images of three different persons from different angles and with different facial expressions is used. To overcome the problem of color constancy and lighting variations in the environment, near-infrared (NIR) imagery is employed. The use of NIR imagery brings a new dimension for applications of invisible lights for computer vision and pattern recognition. This type of imaging hardware device not only provides appropriate active frontal lighting but also minimizes lighting from other sources, making it very suitable for facial feature tracking algorithms such as the AAM. The result of passing a facial region through the active appearance model is an automatically annotated frame with 23 landmark points placed.

Fig. 14 23-feature-point face model.

By analyzing their displacements over multiple frames and relations to empirical thresholds, a characteristic motion pattern for various AUs can be established.

As described in Chapter I, the Facial Action Coding System (FACS) is a human observer-based system designed to describe subtle changes in facial features. When people make faces, whether spontaneous expressions or deliberate contortions, they engage muscles around the eyes, mouth, nose, and forehead. With FACS, Ekman and Friesen detailed each muscle or group of muscles and related it to the change in facial feature it produces, assigning it a specific AU. In total, FACS consists of 44 AUs.

TABLE VII describes how the head and facial AUs that are currently supported are measured. Only those AUs relevant to the facial expressions to be detected are considered. h denotes an empirically determined threshold between tracked AUs and activation of that AU. TABLE VIII lists the nine head and affective cues that are currently supported by the implemented system and their underlying actions. In the computational model of affective states, affective cues (ACs) serve as an intermediate step between measured facial movement and inferred affective state.

TABLE VII AUs with respect to the points in Fig. 24.

AU	Action	Measurement
51/52	Head Turn Left/ Right	$\frac{\text{Dist}(P_7,P_9)}{\text{Dist}(P_{14},P_{12})} > h_t$ or $\frac{\text{Dist}(P_{14},P_{12})}{\text{Dist}(P_7,P_9)} > h_t$
53/54	Head Up/ Down	Vertical disp. of $\left[\frac{1}{2}(P_{17}+P_{18})\right][t,t-1]$
55/56	Head Tilt Left/ Right	$\text{Slope}(P_9,P_{14}) > h_a$ or $\text{Slope}(P_{14},P_9) > h_a$
12	Lip Corner Pull	$\frac{\text{Dist}(P_{19},P_{21})[t]}{\text{Dist}(P_{19},P_{21})[0]} > h_c$
18	Lip Pucker	$\frac{\text{Dist}(P_{19},P_{21})[t]}{\text{Dist}(P_{19},P_{21})[0]} < h_p$
27	Mouth Stretch	$\text{Dist}\left\{\left[\frac{1}{2}(P_{19}+P_{21})\right],P_{23}\right\} > h_{m_l}$ & $\left(\begin{array}{c}\frac{\text{Dist}(P_{20},P_{23})}{\text{Dist}(P_{23},P_{22})} > h_{m_r} \\ \vee \\ \frac{\text{Dist}(P_{23},P_{22})}{\text{Dist}(P_{20},P_{23})} > h_{m_r}\end{array}\right)$
1&2	Eyebrow Raise	$\frac{\text{Dist}(P_1,P_7)[t]}{\text{Dist}(P_1,P_7)[0]} > h_{e_r}$ & $\frac{\text{Dist}(P_3,P_9)[t]}{\text{Dist}(P_3,P_9)[0]} > h_{e_r}$
4	Eyebrow Drop	$\frac{\text{Dist}(P_6,P_{14})[t]}{\text{Dist}(P_6,P_{14})[0]} > h_{e_d}$ & $\frac{\text{Dist}(P_4,P_{12})[t]}{\text{Dist}(P_4,P_{12})[0]} > h_{e_d}$

TABLE VIII List of affective cues and their AUs.

Affective Cue	Description
Head Nod	Alternating AU53 and AU54
Head Shake	Alternating AU51 and AU52
Head Tilt	AU55 or AU56
Head Turn	Pose of AU51 or AU52
Lip Pull	AU12
Lip Pucker	AU18
Mouth Open	AU27
Eyebrow Raise	AU1 and AU2
Eyebrow Drop	AU4

3) Affective Cue Computation

The head nod or head shake is composed of periodic motion that recurs at regular intervals. The temporal structure of a natural head shake or nod is characterized by alternating head-turn-right and head-turn-left or head-up and head-down motion cycles, respectively. The cycle time is the interval of time during which a sequence of recurring motions is completed. An eight-stage state machine is constructed to identify each of these periodic actions over time regardless of the starting direction.

The remaining head and facial displays listed in TABLE VIII are characterized by the AU with which they are associated. The structure of the state machine used to detect these two states is shown in Fig. 15. Since both state machines share the same structure they are both represented in a single figure. Labels in bold correspond with detection of a head nod while labels in italics correspond to the detection of a head shake.

TABLE IX Most likely and discriminative action cues.

Affective State	Most Likely AC	Most Discriminative AC
Agreement	Lip corner pull	Head Nod
Disagreement	Head Shake	Head Shake
Concentrating	Teeth Present	Head Tilt
Interested	Eyebrow Raise, Mouth Open, Teeth Present	Eyebrow Raise, Mouth Open, Teeth Present
Thinking	Head Tilt, Head Turn	Head Tilt, Head Turn
Unsure	Head Turn	Teeth Absent, Mouth Closed

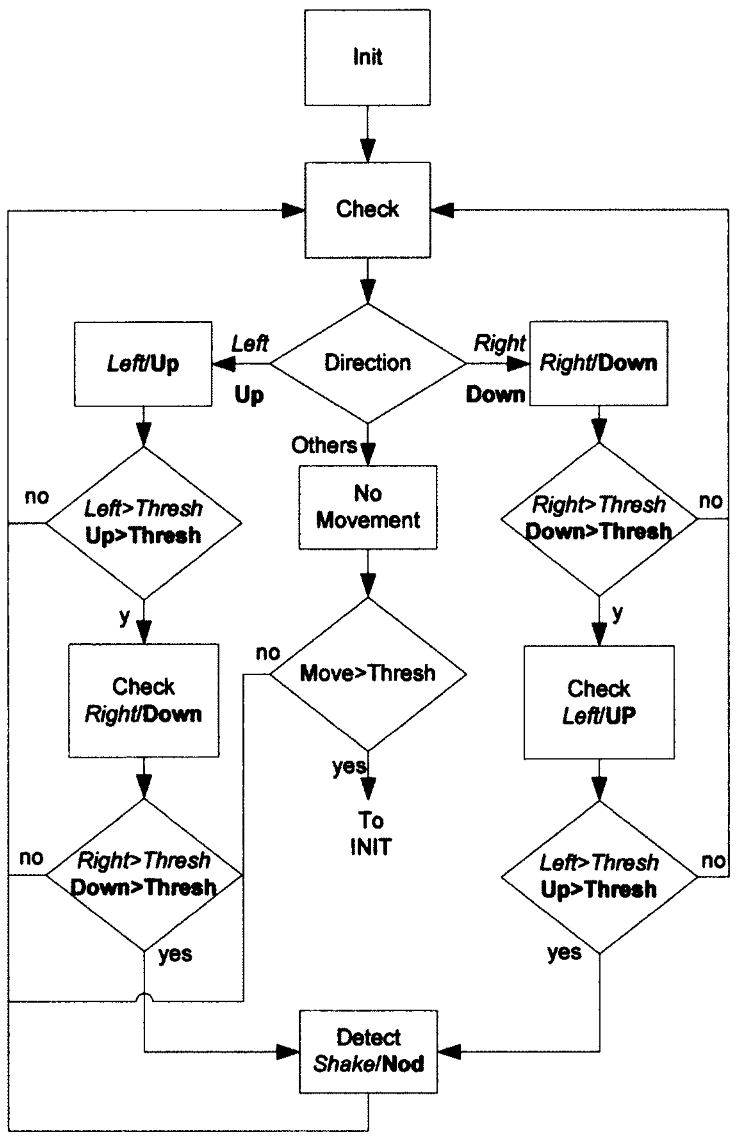


Fig. 15 Head shake/nod detection state flow chart.

4) *Affective State Computation*

The parameters of the dynamic Bayesian networks developed by Kaliouby were trained using statistical information relating probability distribution and discriminative-power heuristics for each display (or AC) and mental state (or affective state) combination [32]. This same information was used to determine which ACs were most likely and most discriminative for each of the affective states to be identified.

TABLE IX presents the simplified findings of Kaliouby relating ACs to affective states.

A binary decision tree structure composed of nine ACs is introduced to improve the performance of Kaliouby's system in terms of frames per second. Note that this paper does not claim that this approach is more accurate in recognizing the user's mental states as compared to the Kaliouby's mind-reading system. The system presented is merely an alternative in which the tradeoff between the computation time and the accuracy of the inference is important. A diagram of the binary decision tree used is presented in Fig. 16.

It is important to understand that the choice of the order of the affective states in the binary decision tree seen is designed in such a way that it prioritizes the affective states desired in an HCI problem. For example, agreement is detected first and disagreement second, as these gestures play an important role in this particular advanced HCI environment.

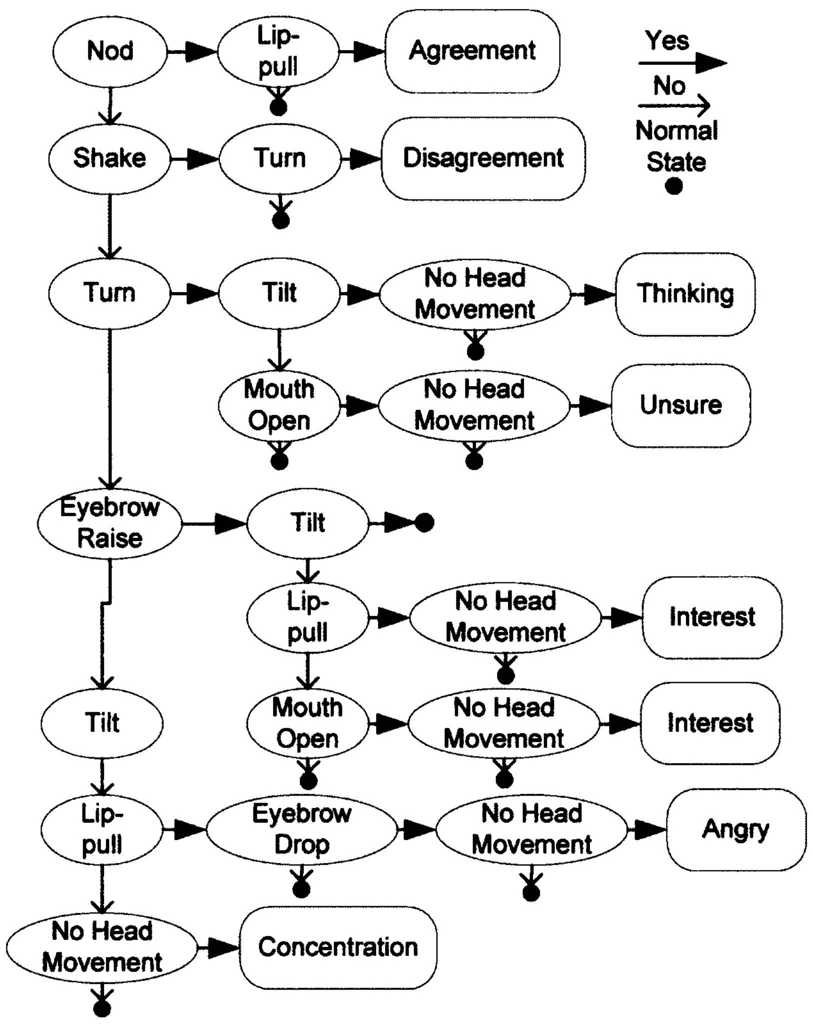


Fig. 16 Binary decision tree structure.

C. Experimental Results

Each phase of the system developed was tested incrementally. An implementation of the full Viola-Jones detector was developed to localize the face in the image. The output of the face detector is an image region containing the face. The face detector successfully detects faces across all sizes. This implementation was tested for video processing on a Pentium IV 2.8GHz machine with 512MB of memory. The face detector can process a 360×240 pixel image in about 0.019 seconds or 52 frames per second.

In order to create more examples and enhance the tolerance to low-resolution images, motions, and variations in intensity, a series of smoothing operations is applied to the initial set of examples. This step teaches the system how to cope with situations where the AAM is fed with a weak, over-smoothed, or poorly-contrasted signal. Finally, the training set reached the number of 3,198 face images. Each training image was hand-annotated with 23 tracking points consistent with Fig. 14.

The tracker performed successfully, tracking at a rate of 38 frames per second with a frame size of 320×240 with minimum misalignment to the face feature for natural human head motion. Typical ranges for head motion are between 70° - 90° of downward pitch, 55° of upward pitch, 70° of turn, and 55° of tilt. Examples are given in Fig. 17.

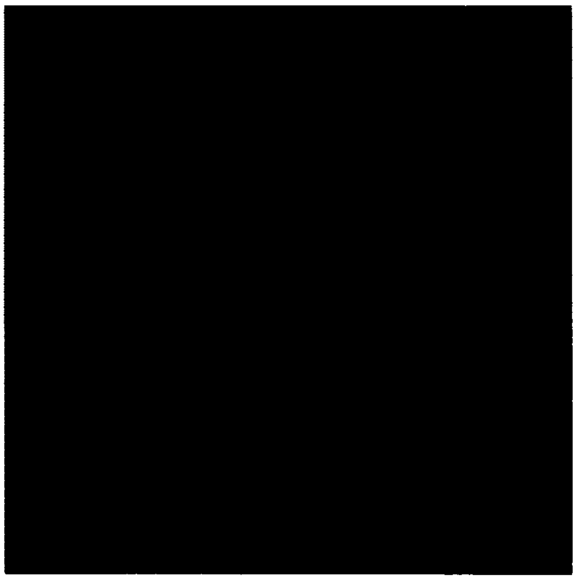


Fig. 17 Facial landmark point tracking.

The complete system ran successfully, detecting the ACs and detecting emotional states in real time. Head nod and head shake detectors functioned successfully. Detection of the head tilt, head turn, lip pull, lip pucker, mouth open, eyebrow raise, and eyebrow drop actions were also performed successfully. Examples of these detections of ACs are presented in Fig. 18.

These ACs served as input to the binary decision tree which successfully detected seven emotional states. Examples of these detections of affective states are presented in Fig. 19.

The binary decision tree-based affective states machine was tested on an Intel Pentium IV, 2.8 GHz processor with 512 MB of memory. The processing time at each of the levels of the system is summarized and compared with Kaliouby's system in

TABLE X. The mind-reading machine developed by Kaliouby was tested on an Intel Pentium IV, 3.4 GHz processor with 2 GB of memory [32]. Even utilizing slower machines, the total processing time for the proposed system is significantly reduced. The resulting frame rate achieved is 36 frames per second.

[redacted] [redacted] [redacted]

Fig. 18 Affective cue detection examples.

[redacted] [redacted]

[redacted] [redacted]

[redacted] [redacted] [redacted]

Fig. 19 Affective states detection examples.

TABLE X System processing time (ms).

Module	Kaliouby [32]	Proposed
Detection/Tracking	3.00	2.60
Action Unit	0.09	0.08
Affective Cue	0.14	0.03
Affective State	41.10	0.06
Total	44.33	2.77

D. Summary

The presented system is capable of recognizing seven affective states: agreeing, concentrating, disagreeing, interested, thinking, unsure, and angry. The ability to detect these seven states shows promise in changing the way an individual communicates with a computer. The real-time system functions in three stages. Viola-Jones face detection is used to identify faces within a frame. An AAM is then used to map 23 landmark points to key locations on the face. Finally, these landmarks are used to form AUs which form ACs as successive frames pass. Depending on which ACs occur, a binary decision tree detects emotional states. The resulting system achieved a real-time frame rate of 36 frames per second.

VII. Feature Point Localization

This chapter will discuss the proposed method for training and operation, using an architectural configuration of a system of nested cascade detectors with the ability to focus in on a feature point of interest.

Each cascade is trained using positive and negative example sets of images resized to a smaller, 24×24 image patch to reduce the size of the possible feature set and decrease the training time. However, when operating on a prospective image frame the detector must be scaled and shifted to detect all possible sizes and positions of the object within the entire frame. The proposed architecture seeks to perform a coarse-to-fine search. The entire object of interest is first identified followed by the identification of regions within the object and the specific features within those regions. The responses of the individual Haar-like features comprising those features at the coarse level are used to trigger the finer-level searches. In this manner each subsequent fine-level search can be triggered as quickly and efficiently as possible.

A. Tree Architecture

Due to its iterative nature a tree structure is selected to relate detection cascades for the proposed point detector. The top of the tree detects the overall object. In the case of the facial feature point location, the face forms the top level. From this "parent" node all the detectors targeted at sub-regions contained within the parent object form the "child" nodes. Fig. 20 outlines the various aspects relating the cascade structure to its train and operation.

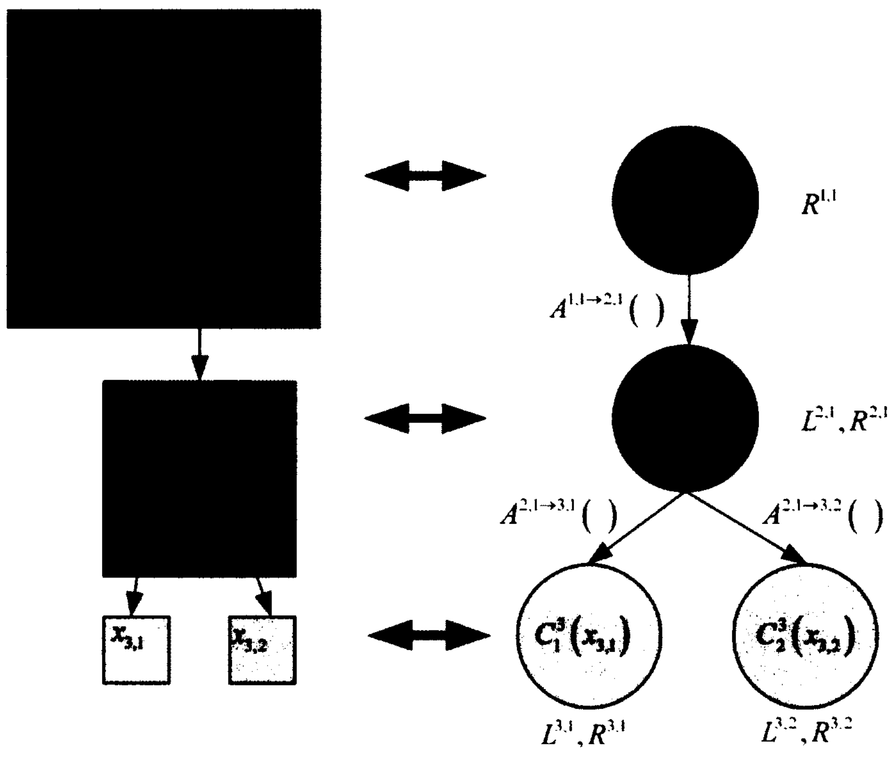


Fig. 20 Point Detection Architectural Structure

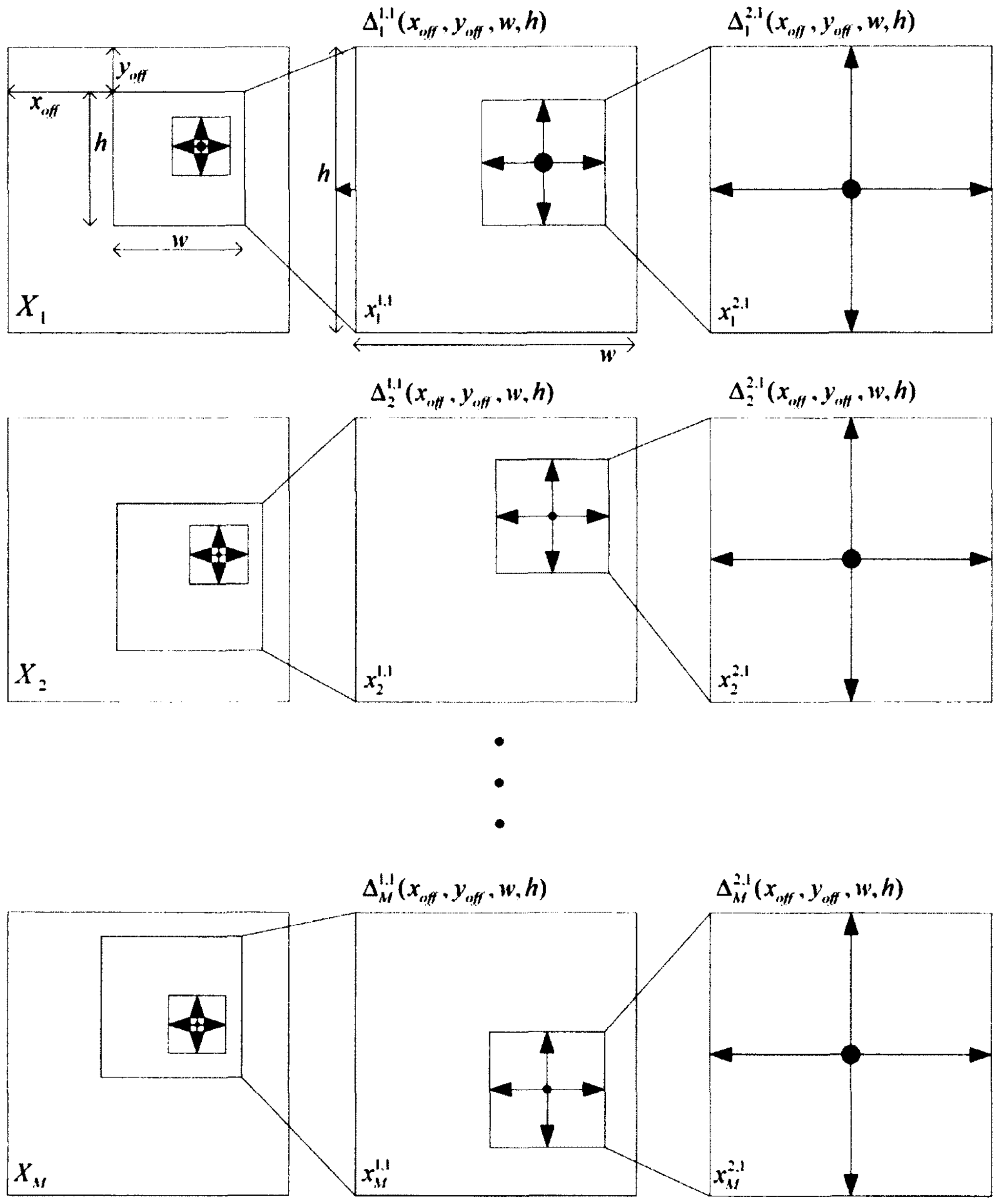
$C_s^l(x)$ is the GentleBoost cascade of Haar-like features trained to detect objects of type $x_{l,s}$, where l is the depth within the detection tree and indexes the sub-region type. $A^{p\rightarrow c}\left(\Phi_p,\tau_{p,c}\right)$ is the activation function for child node c. The activation of the child node c is a function of a subset of the response of features, Φ_p, contained within the parent cascade and an activation threshold, $\tau_{p,c}$. L^c is the initial search location and scale within the region of the parent, R^p.

B. Training Set

The first step in training the cascade is the creation of a training set containing positive examples of each region, sub-region, and feature point to be detected. To create this set each sub-region must be cropped out of its parent region while keeping track of its original position in the original full image. A box equal to the size of the training images is centered over the feature points when cropping images to represent feature points. All sample images are then resized to that specified by the GentleBoost training algorithm. The result is a positive training set $P^{l,s}$ for each node of the following form:

$$P^{l,s} = \left\{\left(X_m^{l,s}, \Delta_m^{l,s}\right)\right\} : m = 1,\ldots,M \tag{57}$$

such that $X_m^{l,s}$ is the resized image patch and $\Delta_m^{l,s}$ can be used to remap the training image back to its initial position and scale in the original uncropped training image. The same negative training set can be used in the training of all the cascades. Fig. 21 illustrates the creation of two layers of an arbitrary positive training set. Each remapping, $\Delta_m^{l,s}$, is a function of x and y offsets as well as a height and width.



$\Delta_1^{1.1}(x_{off}, y_{off}, w, h)$
$\Delta_1^{2.1}(x_{off}, y_{off}, w, h)$
y_{off}
x_{off}
h
w
X_1
$x_1^{1.1}$
$x_1^{2.1}$
$\Delta_2^{1.1}(x_{off}, y_{off}, w, h)$
$\Delta_2^{2.1}(x_{off}, y_{off}, w, h)$
X_2
$x_2^{1.1}$
$x_2^{2.1}$
$\Delta_M^{1.1}(x_{off}, y_{off}, w, h)$
$\Delta_M^{2.1}(x_{off}, y_{off}, w, h)$
X_M
$x_M^{1.1}$
$x_M^{2.1}$

Fig. 21 Training set creation example

C. Activation Functions

Each fine-level search should be activated as soon as the sub-frame represented in the parent node shows promise that it may lead to a positive detection. If at any point the sub-frame is rejected by the parent node, the operation of all child nodes is also terminated. When the coarse-level search detects features similar to those present in the child cascade the child node should begin its search. The goal of the activation function is the identification of any possible correlation between the locations of coarse and fine-level features. An under-laying facial structure can be learned based on the response of the Haar-like feature present in the cascades.

One advantage of a nested cascade structure is that the order in which features are selected in the cascade can be directly tied to their importance as representative components of the overall object. Each cascade can be defined as having an ordered feature set while excluding weak classifiers which are representative of entire layers of the cascade.

$$F^{l,s} = \left\{ f_1^1, \ldots, f_T^1, f_2^2, \ldots, f_T^2, f_2^j, \ldots, f_T^j \right\} \tag{58}$$

Haar-like features must be remapped to the training set to locate features which are spatially related to one another at multiple scales. Each feature must be further broken down so that all possible edges of each feature are examined.

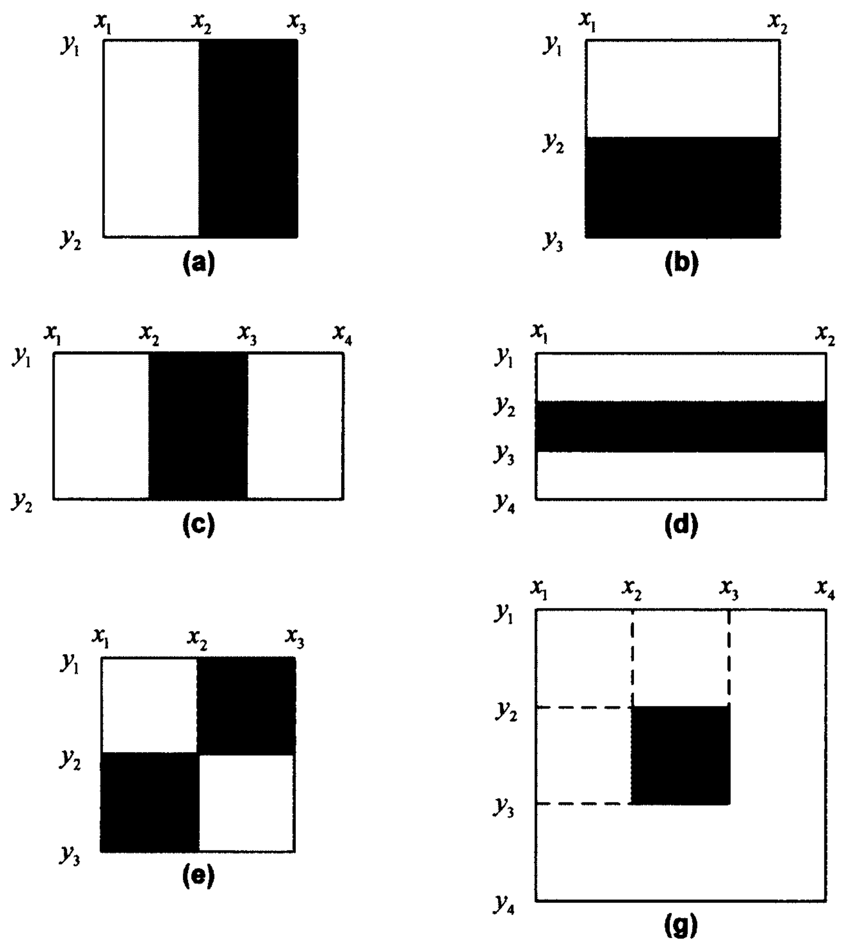


Fig. 22 Haar-like feature with edge labels.

Each feature is assigned a varied set of edges depending of the feature type. These are defined by

$$\chi_m^f = \{x_1,\dots,x_N\},\{y_1,\dots,y_N\} : N_x, N_y \in \{2,3,4\} \tag{59}$$

where f references the feature and m refers to the index of the training image. These values can be referenced by $\chi_m^{f,x}(n) = x_n$ and $\chi_m^{f,y}(n) = y_n$. A breakdown of the edge set for each type of features is shown in Fig. 22.

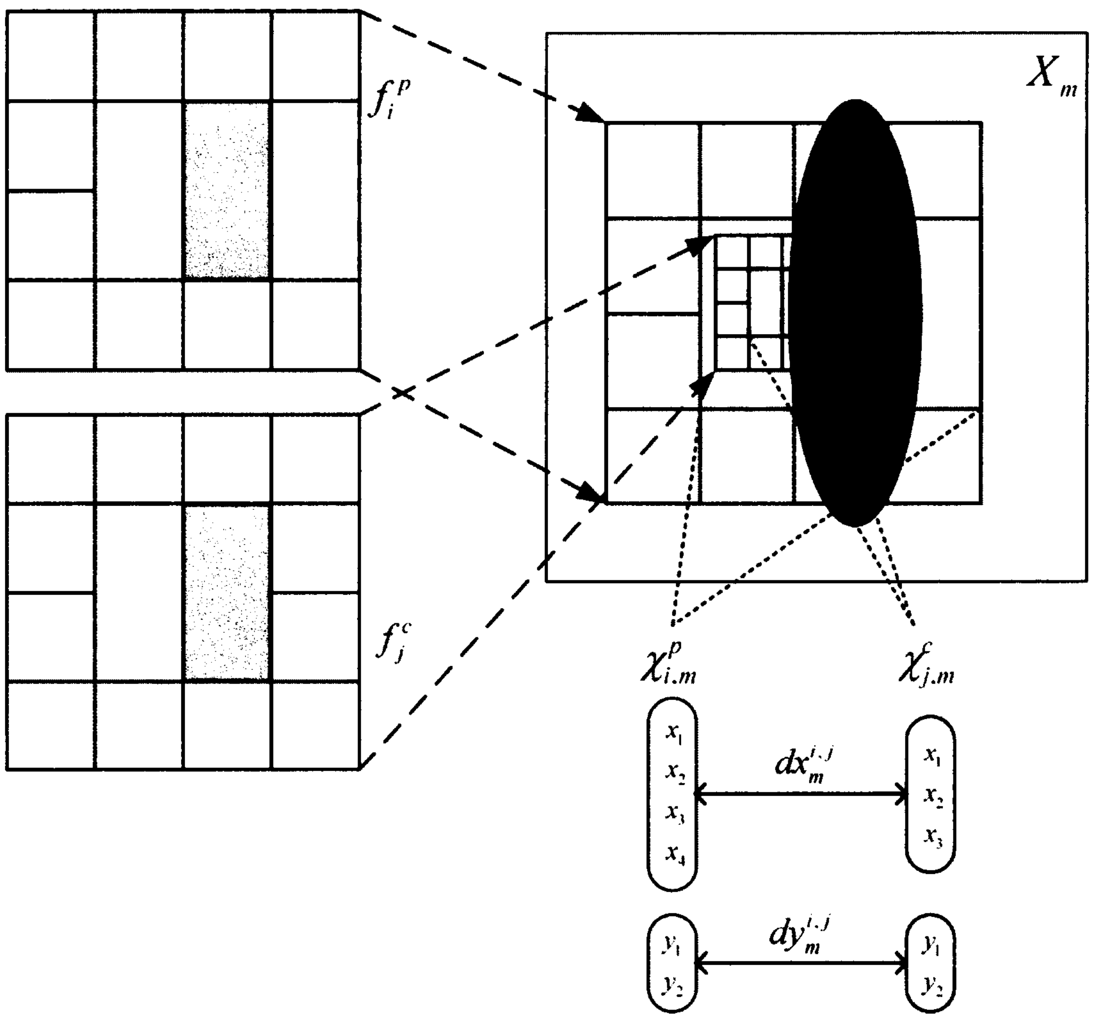


Fig. 23 Activation function calculation example.

The following procedure is used to locate the activation function, $A^{p\to c}$, from parent node p to child node c. First, the feature sets for each cascade must be found and indexed so that F_i^p represents the i^{th} weak classifier in the parent cascade and F_i^p represents the j^{th} weak classifier in the child cascade. For each training image, X_m, one feature from both the parent cascade and the child cascade is mapped back to the training image using $\Delta_m^{l,s}$. The locations of all feature edges on the test image are collected in sets $\chi_{i,m}^p$ and $\chi_{j,m}^c$.

$$f_i^p \overset{\Delta_m^p}{\rightarrow} X_m : \Delta_m^p\left(f_i^p\right) = \chi_{i,m}^p \tag{60}$$

$$f_i^c \overset{\Delta_m^c}{\rightarrow} X_m : \Delta_m^c\left(f_i^c\right) = \chi_{i,m}^c \tag{61}$$

The distances between the parent and child edge locations are then calculated in both the horizontal and vertical directions.

$$dx_m^{i,j}(n_i, n_j) = \left|\chi_{i,m}^{p,x}(n_i) - \chi_{j,m}^{c,x}(n_j)\right| \tag{62}$$

$$dy_m^{i,j}(n_i, n_j) = \left|\chi_{i,m}^{p,y}(n_i) - \chi_{j,m}^{c,y}(n_j)\right| \tag{63}$$

The mean value of each of these edge differences is found over the entire training set to find $\overline{dx^{i,j}}\left(n_i, n_j\right)$ and $\overline{dy^{i,j}}\left(n_i, n_j\right)$.

Average edge differences are calculated between the first N_p features in the parent cascade and the first N_c features in the child cascade. The N_a parent features with the lowest average distance are then chosen to be added to the activation function feature set.

$$\Phi^{p\rightarrow c} = \left\{f_t(x)\right\} : t = 1, \ldots, N_a \tag{64}$$

The test set from the parent cascade, X^p, is used to determine an activation threshold $\tau^{p\rightarrow c}$.

$$\tau^{p\rightarrow c} = \min\left(\sum_{t=1}^{N_a} F_t\left(X_m^p\right)\right) : m = 1, \ldots, M \tag{65}$$

The final form of the activation equation is

$$A^{p\rightarrow c}(X)=\begin{cases}1 & \sum_{t=1}^{N_a} f_t(X)\geq \tau^{p\rightarrow c} : f_t \in \Phi^{p\rightarrow c} \\ 0 & otherwise\end{cases} \tag{66}$$

An equation of this form must be found for each connection between parent and child nodes. The advantage to this approach is that the feature values need not be recomputed as their values can be extracted from the parent cascade. As soon as an activation value of one is produced the child node begins its search.

D. Search Regions

Once a node is active it must begin searching a region of the image. The head node begins searching the prospective sub-frame using a coarse-to-fine scan of the image. As soon as a prospective object begins to be identified there is a chance that one or several of its child nodes may become active. If this does occur one approach would be to search within the entire prospective object's region of the sub-frame in the same manner as the initial search. Because of the structure of the tree and the method used to create the dataset this approach would be valid. However, the presence of some underlying consistent structure could significantly reduce the necessary search area, leading to higher efficiency.

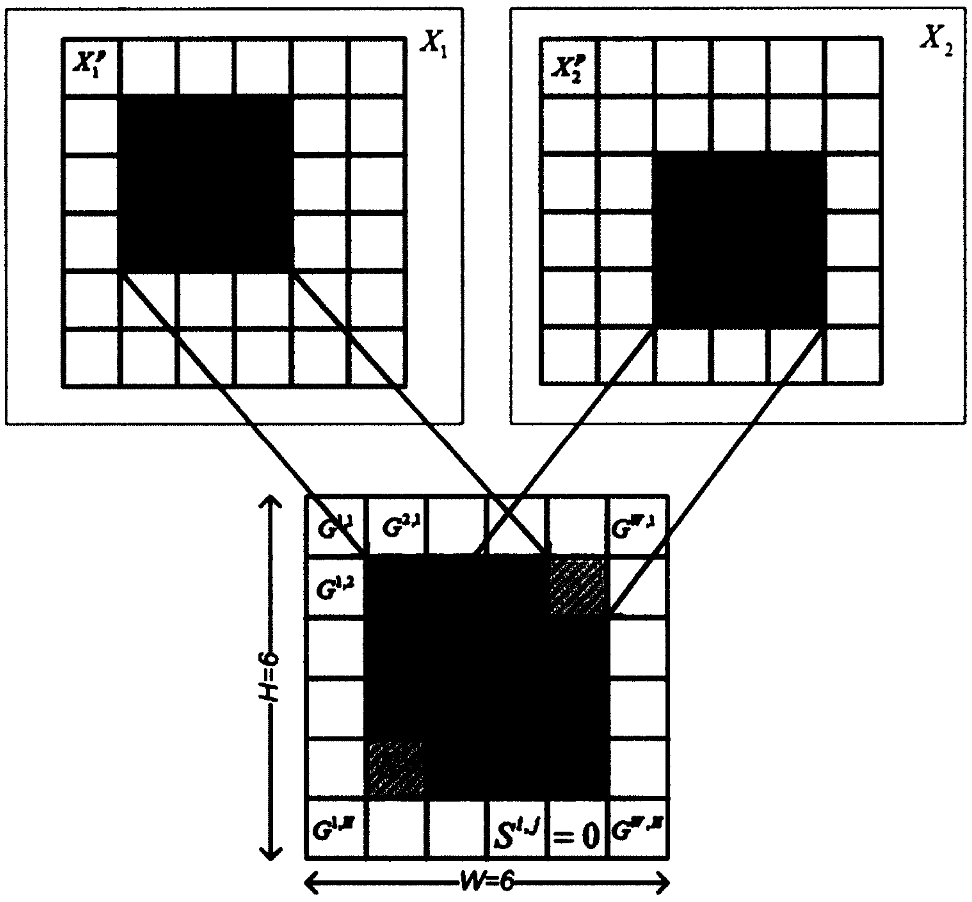


Fig. 24 Search region generation example.

The training set can be used to reduce this search area by using a remapping technique similar to that employed by the training of activation functions. First, a $W \times H$ grid of square regions, $G^{i,j}$, is defined such that the dimensions are equivalent to the size of the images used to train the cascades.

$$G^{i,j} : i \in \{1,\ldots,W\}, j \in \{1,\ldots,H\} \tag{67}$$

The grid forms image segments equivalent to one pixel in the training image and the smallest area which can be summed during the operation of the detector. For each training image the grid is mapped to X_m^p which is then mapped to the overall image, X_m. X_m^c is also mapped to X_m. Any square within $G^{i,j}$ which overlaps the region covered by X_m^c is marked as an area which must be searched, $S_m^{i,j}$.

$$S_m^{i,j} = \begin{cases} 1 & R\left(G^{i,j} \to X_m^p \to X_m\right) \cap R\left(X_m^c \to X_m\right) \neq \varnothing \\ 0 & otherwise \end{cases} \tag{68}$$

Such an area is found for each example in the training set. The composite search area is found by combining all of the individual results.

$$S^{i,j} = \begin{cases} 0 & \sum_m =1^M S_m^{i,j} = 0 \\ 1 & \sum_m =1^M S_m^{i,j} \neq 0 \end{cases} \tag{69}$$

The final search region, $Sr_p^{i,j}$ is found by the smallest rectangle which can completely cover each region for which $S^{i,j} = 1$. Only these areas of the parent sub-frame need to be searched for possible instances of the child object. By using this technique the search for a landmark feature point is reduced by incrementally zooming into the area of importance at relatively low computational cost at run time. Since each search is triggered by the features output from the coarse scale search above the current level there is little operational cost. Also by using training data to isolate the most likely search region the amount of area that must be parsed the fine-level detector is also minimized. The result is that the time to search for a specific point is the time it takes for each layer to locate its intermediate sub-image and trigger the finer level search for each node hit on the way down the tree.

VIII. DATABASE GENERATION FOR AN AUTOMATED FACIAL MOVEMENT TRACKING SYSTEM

The detection tree architecture described in Chapter VII is applied to the problem of automated facial movement tracking in this proposal. This section will discuss the face model and the method used to create the training and evaluation images. The structure of the overall architecture as well as the development of a testing procedure will also be presented.

A. Face Model

A 26-point face model is applied to each candidate facial image frame as shown in Fig. 25. In this model three points are taken for each eyebrow, five points per eye, four nose points, four mouth points, and two chin and jaw points. With accurate location of these points it should be possible to detect up to 38 facial AUs and their combinations. Points corresponding to the inner eye corners are of utmost importance since these are the only points which are stationary with respect to the face's position over the skull. These points can be used for normalization and parameterization of the facial structure and current state.

B. Training Set

An extensive training set is needed to accurately train a detector system as described in Chapter IV. To build this training set the Cohn-Kanade Facial Expression

Database [70] is used. This image sequence database consists of approximately 500 image sequences from 100 subjects and is accompanied by metadata including AUs and emotion specific expressions. A subset of this dataset consisting of multiple individuals making a variety of expressions was processed by a specially created GUI Matlab program. The use of this program allowed for the manual cropping of specific facial regions and features while automatically naming and saving resulting files. The face region forms the head node of the tree and is connected to four child nodes comprising the right and left upper facial regions, a nose region, and a lower face region. Each of the upper and lower face regions contains two sub-regions to more closely focus on regions of interest which should contain feature points. At the bottommost layer of each branch is the feature point detection shown as a rectangle.

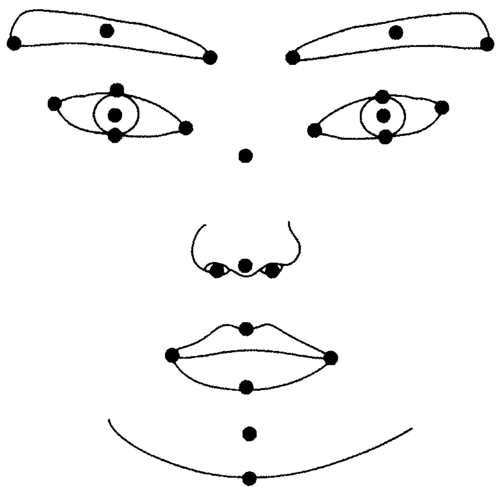


Fig. 25 26-point face model.

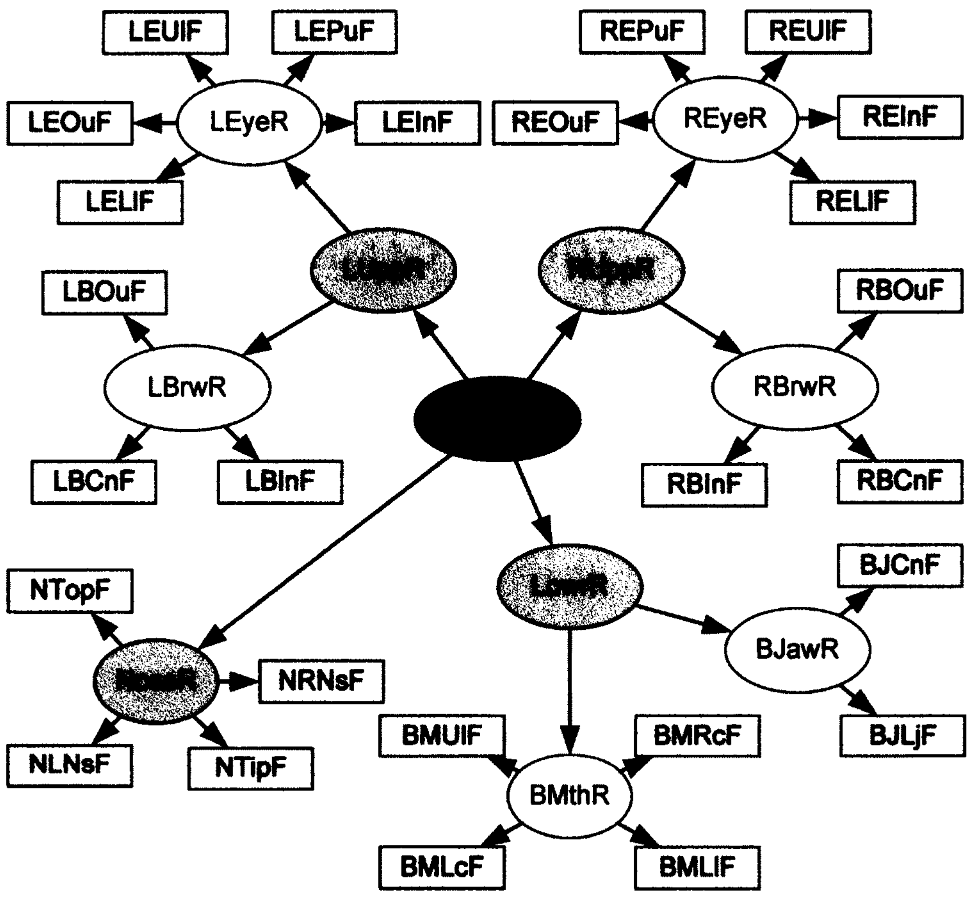
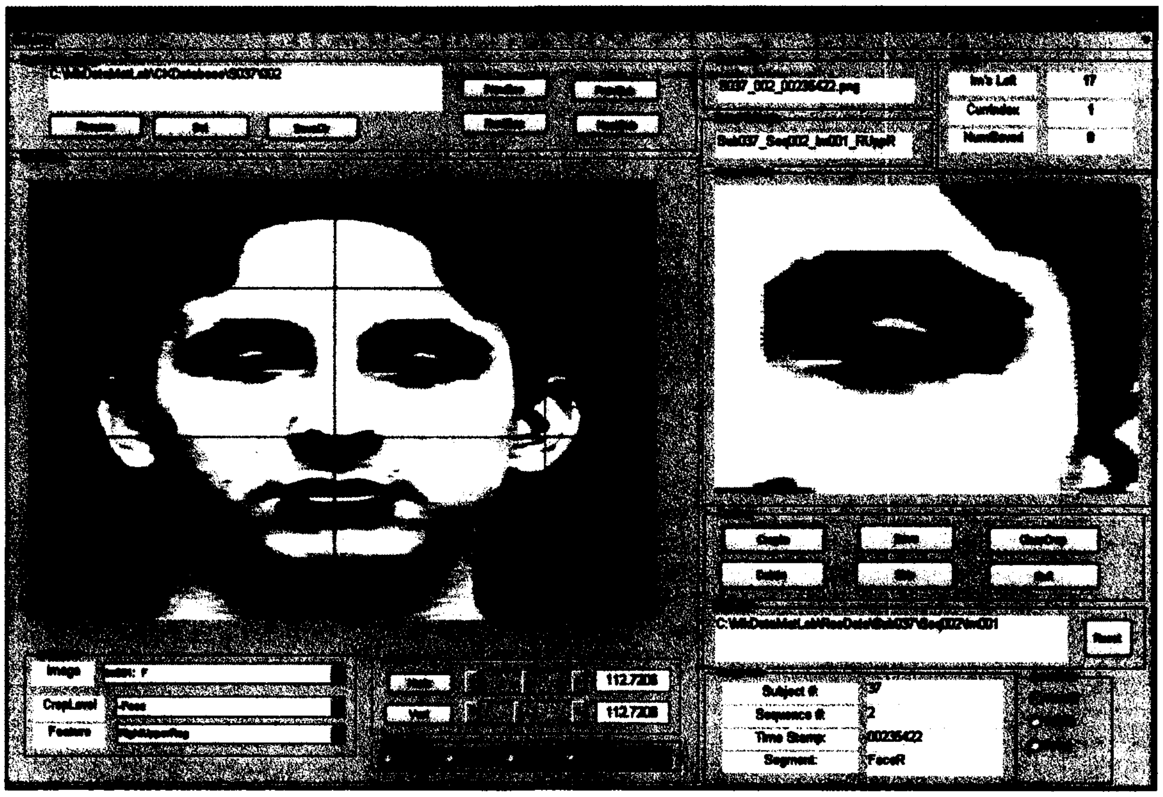


Fig. 26 Facial feature point localization architecture.

Information relating to the position within the original frame was also maintained so as to allow for the remapping during training as required by Sections VII.C and VII.D as well as for use as ground truths in the testing process. A screen shot of this program can be seen in Fig. 27.



S037_002_00235422.png

Sub037_Seq002_Im001_RUppR

Im's Left	17
CurIndex	1
NumSaved	8

Subject #:	37
Sequence #:	2
Time Stamp:	00235422
Segment:	FaceR

Fig. 27 GUI screen shot.

Each specific region and feature has a specific set of instructions detailing the creation of each sub-image (presented in TABLE XI - TABLE XVII). The facial region is first cropped from the training image and then divided into four general sub-regions (see TABLE XI). The result of each region is a square sub-frame with the exception of those regions and features marked with an asterisk. During training these regions will be scaled down to an empirically determined mean aspect ratio. The upper and lower sub-regions are then further divided into targeted regions of interest (see TABLE XII).

TABLE XI Face and facial sub-regions.

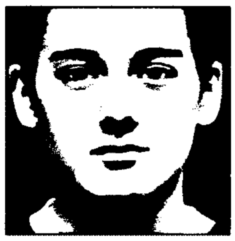
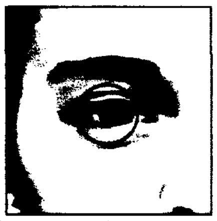
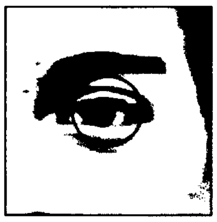
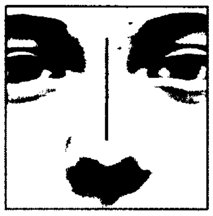
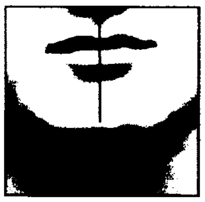
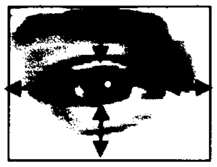
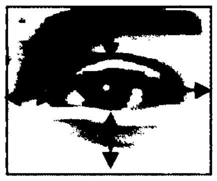
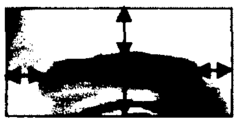
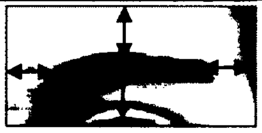
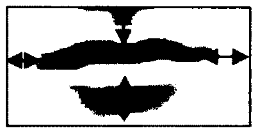
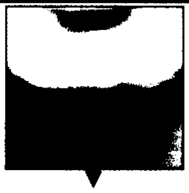
Region: Name(ImageCode)	Instructions	
	Parent Node	Child Nodes
 Face Region: FaceR(F)	1. Left Face Edge 2. Right Face Edge 3. Middle of Forehead 4. Halfway between Chin and Bottom of Neck 5. Square Horizontally 6. Center Horizontally on the Nose	
	Full Frame	Left Upper Region: LUppR Right Upper Region: RUppR Nose Region: NoseR Lower Face Region: LowrR
 Left Upper Region: LUppR(Lu)	1. Center of Ear or edge of Face 2. Center of Nose 3. Center of Forehead 4. Center of Cheek 5. Square Vertically 6. Center Vertically to the Pupil of the Eye 7. Adjust so Outer Brow is still well Inside Left Edge and Nose Center is still near Right Edge.	
	Face Region: FaceR	Left Eye Region: LEyeR Left Brow Region: LBrwR
 Right Upper Region: RUppR(Ru)	1. Center Nose 2. Center of Ear or Edge Face 3. Center of Forehead 4. Center of Cheek 5. Square Vertically 6. Center Vertically to the Pupil of the Eye 7. Adjust so Outer Brow is still well Inside Left Edge and Nose Center is still near Right Edge.	
	Face Region: FaceR	Right Eye Region: REyeR Right Brow Region: RBrwR
 Nose Region: NoseR(N)	1. Inner Left Cheek 2. Inner Right Cheek 3. Lower Forehead 4. Upper Lip 5. Square Horizontally 6. Center Nose Horizontally	
	FaceRegion: FaceR	Nose Left Nostril Feature: NLNsF Nose Right Nostril Feature: NRNsF Nose Tip Feature: NTipF Nose Top Feature: NTopF
 Lower Face Region: LowrR(L)*	1. Halfway between Left Lip Corner and Left Face Edge 2. Halfway between Right Lip Corner and Right Face Edge 3. Below Nose 4. Bottom of Frame 5. Center Horizontally on Mouth	
	FaceRegion: FaceR	Mouth Region: BMthR Jaw Region: BJawR

TABLE XII Targeted sub-regions.

Region: Name(ImageCode)	Instructions: Parent Node	Instructions: Child Nodes
 Left Eye Region: LEyeR(E)*	1. >12 Pixels Left of Outer Corner 2. >12 Pixels Right of Inner Corner 3. Above Eyebrow 4. Below "Bags" of Eyes 5. Center on Pupil 6. Adjust Horizontally so 1&2 are met	
	Left Upper Region: LUppR	Left Outer Corner Feature: LEOuF Left Upper Lid Feature: LEUlF Left Eye Inner Corner Feature: LEInF Left Eye Lower Lid Feature: LELlF Left Eye Pupil Feature: LEPuF
 Right Eye Region: REyeR(E)*	1. >12 Pixels Left of Inner Corner 2. >12 Pixels Right of Outer Corner 3. Top of Brow 4. Below "Bags" of Eyes 5. Center on Pupil 6. Adjust Horizontally so 1&2 are met	
	Right Upper Region: RUppR	Right Outer Corner Feature: REOuF Right Upper Lid Feature: REUlF Right Eye Inner Corner Feature: REInF Right Eye Lower Lid Feature: RELlF Right Eye Pupil Feature: REPuF
 Left Brow Region: LBrwR(B)*	1. >12 Pixels Left of Outer Corner 2. >12 Pixels Right of Inner Corner 3. >12 Pixels Above Center of Uppermost Arc of Eyebrow 4. >12 Pixels Below Lowest of Inner and Outer Corner	
	Left Upper Region: LUppR	Left Brow Outer Corner Feature: LBOuF Left Brow Inner Corner Feature: LBInF Left Brow Center Feature: LBCnF
 Right Brow Region: RBrwR(B)*	1. >12 Pixels Left of Inner Corner 2. >12 Pixels Right of Outer Corner 3. >12 Pixels Above Center of Uppermost Arc of Eyebrow 4. >12 Pixels Below Lowest of Inner and Outer Corner	
	Right Upper Region: RUppR	Right Brow Outer Corner Feature: RBOuF Right Brow Inner Corner Feature: RBInF Right Brow Center Feature: RBCnF
 Mouth Region: BMthR(M)*	1. >12 Pixels Left of Left Mouth Corner 2. >12 Pixels Right of Right Mouth Corner 3. >12 Pixels Above Center of 'v' at Top of Mouth 4. Bottom of Dark Region Below Lower Lip	
	Lower Region: LowR	Mouth Left Corner Feature: BMLcF Mouth Upper Lip Feature: BMUlF Mouth Right Corner Feature: BMRcF Mouth Lower Lip Feature: BMLlF
 Jaw Region: BJawR(J)*	1. Halfway between Left Cheek and Left Mouth Corner 2. Halfway between Right Mouth Corner and Right Cheek 3. Bottom Third of Dark Region Below Mouth 4. Halfway between Jaw Line and Bottom of Neck (>12 Pixels Below Jaw Line)	
	Lower Region: LowR	Jaw Chin Feature: BJCnF Lower Jaw Feature: BJLjF

With all general regions of interest acquired, image patches are centered on eye, brow, nose, mouth, and chin features. Each image is a square, 24×24 section centered on the specific feature point with the exception of chin and lower jaw features.

TABLE XIII Eye features (children of LEyeR and REyeR).

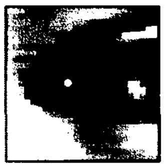
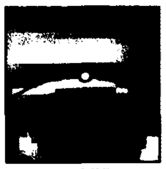
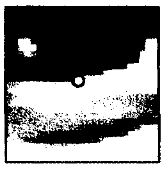
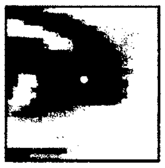
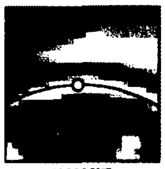
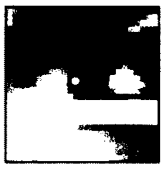
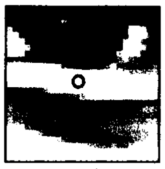
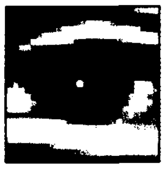
Left Outer Corner  LEOuF	Left Upper Lid  LEUlF	Left Inner Corner LEInF	Left Lower Lid  LELlF	Left Eye Pupil LEPuF
Right Outer Corner  REOuF	Right Upper Lid  REUlF	Right Inner Corner  REInF	Right Lower Lid  RELlF	Right Eye Pupil  REPuF
24x24 Window Centered on Outer Corner of Eye	24x24 Window Centered on Upper Lid Bulge at Upper Most Apex of Arch	24x24 Window Centered on Inner Corner of Eye	24x24 Window Centered on Lower Lid Upper Edge	24x24 Window Centered on Pupil (if not visible skip feature)

TABLE XIV Brow features (children of LBrwR and RBrwR).

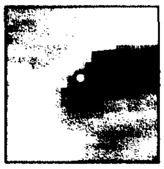
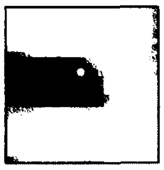
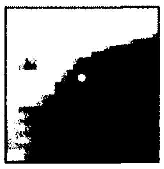
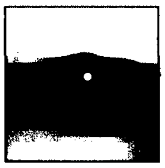
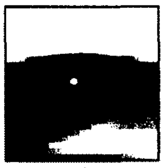
Left Outer Corner  LBOuF	Right Outer Corner  RBOuF	24x24 Window Centered on Outer Corner of Brow (End of the dark corner created by the hair of the brow not necessarily the shadow created by the bone structure)
Left Inner Corner LBInF	Right Inner Corner  RBInF	24x24 Window Center on Inner Corner of Brow (Edge of dark region created by the hair of the brow not necessarily the shadow created by a wrinkle caused by a furrowing of the brow)
Left Center  LBCnF	Right Center  RBCnF	24x24 Window Centered on Upper Arc of Brow in the Middle of the Brow

TABLE XV Nose features (children of NoseR).

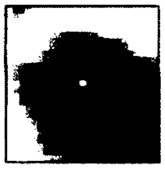
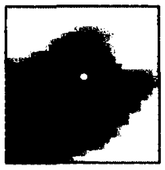
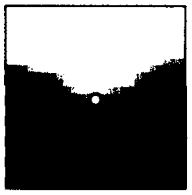
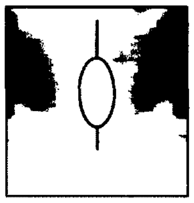
Left Nostril	Right Nostril	Tip	Top
 NLNsF	 NRNsF	 NTipF	 NTopF
24x24 Window Centered on Dark Area of the Right Nostril	24x24 Window Centered on Dark Area of the Right Nostril	24x24 Window Centered on Gradient Change between Light and Dark Vertically and along the Bridge of the Nose Horizontally	1. Inner Corner of Left Eye 2. Inner Corner of Right Eye 3. Bottom of Forehead (near top of Eyebrows) 4. Below Bags of Eyes (Midway down Bridge of Nose) 5. Center Horizontally along Bridge on Nose 6. Square Horizontally

TABLE XVI Mouth features (children of BMthR).

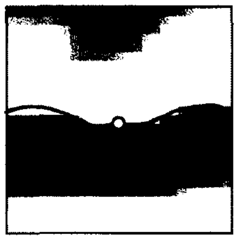
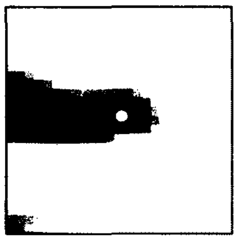
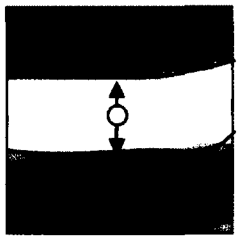
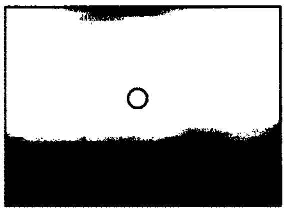
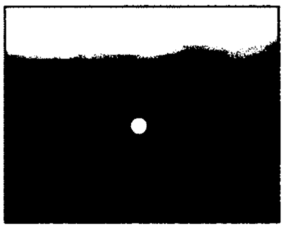
Left Corner	Upper Lip	Right Corner	Lower Lip
BMLcF	 BMUIF	 BMRcF	 BMLIF
24x24 Window Centered on Left Corner (where upper and lower lips meet)	24x24 Window Centered on Bottom of 'v' in Upper Lip	24x24 Window Centered on Right Corner (where upper and lower lips meet)	24x24 Window Centered Horizontally Midway on Lower Lip and Vertically Midway on Lower Lip

TABLE XVII Chin and jaw features (children of BJawR).

Chin  BJCnF	1. Left Edge of Mouth 2. Right Edge of Mouth 3. Just Above Bottom Edge of Dark Region Below Mouth 4. Just Below Dark Region Created by Jaw Line 5. Center on Middle of the Front of the Chin if necessary
Jaw Line  BJLjF	1. Left Edge of Mouth 2. Right Edge of Mouth 3. Halfway between Dark Region Below Mouth and Jaw Line 4. Halfway between Jaw Line and Bottom of Neck 5. Center on Bottom of Jaw Line

Thirteen individuals were selected from the overall dataset and multiple images were cropped from sequences depicting several emotional states. A total of 982 total images of each facial region were created. The format of the filename includes a five-character tag, a six-character subject id, a six-character sequence id, and a five-character image id. For example, "FaceR_Sub010_Seq001_Im001.png" would represent the Face Region of the first image in the first sequence from subject number 10.

During the sampling of sub-frames from the raw training set the x and y offsets, as well as the widths and the height of each sub-frame in relation to its parent image, were maintained in a separate file. A field was added to each image file in the format "_x####_X####_y####_Y####" between the current file name and ".png" portion of the file name. The "_x" and "_X" portions of the code correspond to the minimum and maximum x values respectively. Likewise, the "_y" and "_Y" portions of the code correspond to the minimum and maximum y values of a sub-frame respectively.

Each set of images must be further processed. First, each image is subjected to variance normalized as follows:

$$I_n[x] = 12\left[\frac{I[x]-\mu}{3\sigma}+1\right] \tag{70}$$

The resulting image is then resized to 24×24 for most image segments. For segments which are not square an average ratio between the width and the height was found over those sets of images. This ratio is then expanded to create a region with similar area to that of the square region while most closely emulating the height-to-width ratio.

In addition to the set of positive images for each region, a significantly large set of negative images must also be created and maintained. To generate the negative images, set image segments were randomly selected from the Caltech-256 Object Categories Database [71]. Images which contained faces were manually eliminated before their inclusion in the overall set. A set of 10,000,000 sub-frames in a variety or original ratios was created and stored in a two-layer system of paged directories to allow for faster access to individual files.

The files "CreateNeg.cpp" and "NegIm.h" contain the source code and structures used to create this information. This code takes as an input the source folder which contains the raw images for the Caltech Database, the total number of segments to be generated from each image, the desired sub-frame dimensions, and the destination folder. Due to the large quantity of negative images, storing all resulting images in one directory presented system indexing problems. As a result a paging system was used in its place.

One hundred directories (P1_0-P1_99) were created, each containing 250 directories (P2_0-P2_249). Each directory then contained 400 jpeg segments for each one of the 250,000 images used to create the negative training set.

Functions from the OpenCV Computer vision library were used to import the original raw image and convert that image to its grayscale equivalent. A random segment was then cropped from the raw image. This segment was selected in a manner in which the image segment maintained the same ratio of height to width as the desired segment while also staying within bounds of the raw image's dimension. The cropped image was then variance-normalized in the same manner as the positive set. The resulting image is then resized to the desired dimensions and saved, incorporating the original file name and the original location of the minimum and maximum values of both x and y coordinates.

C. *Preprocessing Training Images*

The first stage of development of a new boosted detector was the conversion of all training images into their integral image form. To accomplish the conversion image transformation, each image is imported as an *IplImage* and then converted into a pixel array of integers ($I(x,y)$). A sum image array ($S(x,y)$), one column wider and one row higher than the original image, is initialized to all zeros. A *RowSum* quantity is also initialized to zero. Pixels are then read moving horizontally across each row. The following algorithm developed by Leinhart [53] is used to compute the integral image for each training image. This process is shown in Fig. 28. Each image is then saved as a raw data file under the same name as the training image with the exception of the ".dat" file extension.

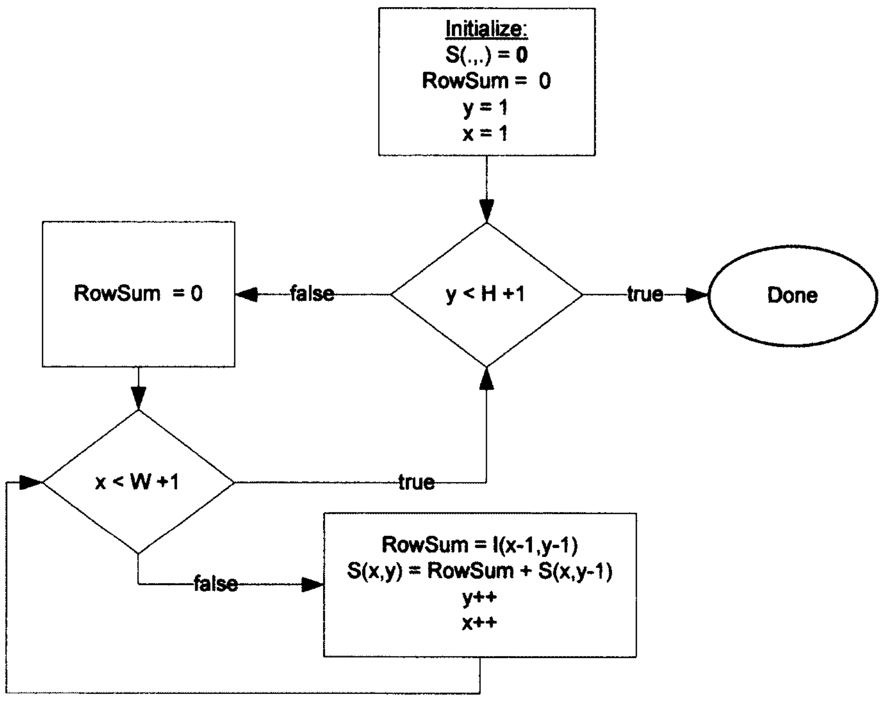


Fig. 28 Integral image computation.

With each raw training image normalized and converted to a properly named integral image, the next logical step was the development of a data structure which could import all of this information and convert it to a format useful to the training program. A single class structure entitled *TrnImage* was built to contain both the positive and negative images. Three sets of fields were extracted and maintained: common fields, positive-only fields, and negative fields.

TABLE XVIII *TrnImage* fields.

Common Fields	Positive Image Fields	Negative Image Fields
ImType{UND, POS, NEG}	*SubName*—Subject Name	*IdName*—Image Name
MapRect{.x,.y, .width, .height}	*SeqName*—Sequence Name	*LocName*—P1_#/P2_#
TypeName{'.png', '.jpg'}	*ImName*—Image Name	*LocNum*— {Page1 Number[0-99], Page2 Number[0-249], Index Number[0-399]}
Name—Complete Name	*SubNum*—Subject Number	
SDim{.width, .height}	*SeqNum*—Sequence Number	
SIm—Integral Image	*ImNum*—Image Number	
Flag—Internal Use		

A two-stage process is used to import each image. For positive training samples the *Subject*, *Sequence*, and *Image* information as well as establishing the *Mapping Rectangle* from information contained within the filename. If the image is a negative training image, two additional fields used to maintain the location in the paged negative image storage system were also extracted in two formats: a string (e.g. P1_1/P2_1/) and a three-value integer vector (e.g. {1,1,1}). The integer vector is used to maintain a record of which images have already been used in the training process so images are not included more than once. Once the names of the images are stored in the structure a second function is called to import the integral image matrix.

The ability to evaluate a Haar-like feature was also incorporated into this class through the implementation of a function entitled *EvalHaar(HaarType, CvRect)*. This function takes in a feature in terms of its type {*V_EDGE*, *H_EDGE*, *V_LINE*, *H_LINE*, *D_LINE*, *C_SURR*} and its location {*X_Offset*, *Y_Offset*, *Width*, *Height*}. The feature is first checked to see if it is within acceptable dimensions of the integral image. If the image is within acceptable dimensions then the feature value is calculated according to TABLE I, TABLE II, and TABLE III.

D. Training Set Definitions

Several sets must be generated and maintained in order to successfully train a detection cascade. The first set which must be generated is the set of positive training images and a set of evaluation images. Two methods of selecting a positive image set were implemented. The first method imports *MaxSize* random images and divides the overall set into two sets. A percentage as defined by *PctEvl* is apportioned to the evaluation set (*EvlSet*) and the remaining images are added to the positive set (*PosSet*). If a value of zero is sent to *MaxSize* then all training images in the specified directory are included in the datasets.

While neither the positive training set nor the evaluation set change, the negative set is continually updated. The first step to handling the negative set is the creation of a list of all possible image locations. The class *NegLocs* is used to maintain a list of three integer vectors which goes from {0,0,0} to {99,249,399}. Each time a negative image is required a set of 10 random combinations is created. These ten image locations are parsed over the list of *NegLocs* until a match is found. If all 10 cases have been used then 10 more options are created and the process is repeated until an image is selected. Once an image has been used once it is removed from the *NegLocs* list of valid images. A negative training set, *NegSet*, of size *MaxSize* is generated randomly by the *GenNegSet* function. If a single image is required the *GetNegImage* function is called and uses the same approach described above.

The *FeatSet* is a list of every possible type of Haar-like feature which can occur within a given training image dimension. For each feature type a unique set of features covering all possible offsets and scales must be added to this list. Each set is defined by the following equation:

$$\left\{\left(FeatType, F_x, F_y, FB_x \times S_x, FB_y \times S_y\right)\right\} : \begin{array}{c} \forall S_x = \left[1, \left\lfloor \dfrac{W}{FB_x} \right\rfloor\right); \\ \forall S_y = \left[1, \left\lfloor \dfrac{H}{FB_y} \right\rfloor\right); \\ \forall F_x = \left[0, W - \left(S_x \times FB_x\right)\right); \\ \forall F_y = \left[0, H - \left(S_y \times FB_y\right)\right); \end{array} \tag{71}$$

In (71) the *FeatType* can refer to any of the six possible Haar-like features discussed earlier. F_x and F_y are the offset from the origins in the x and y directions respectively. FB_x and FB_y are the smallest dimensions possible for each feature in both the x and y directions. For example, for the vertical line feature FB_x would equal three and FB_y could equal one. S_x and S_y are the scaling factors in both directions. W and H are the width and height of the training frame.

TABLE XIX Feature total (24×24).

Feature Type	Total Possible Features
V_EDGE (Vertical Edge)	43,200
V_LINE (Vertical Line)	43,200
H_EDGE (Horizontal Edge)	43,200
H_LINE (Horizontal Line)	43,200
D_LINE (Diagonal Line)	20,736
C_SURR (Center Surround)	8,464
All Features	170,800

E. GentleBoost Training

An initial attempt was made to implement the GentleBoost training algorithm as shown in [52]. To begin the implementation of this algorithm a classifier had to be created to interpret the response of a particular Haar-like feature. Each stump classifier divided the range of possible responses into a fixed number of bins. For each bin the range was defined as well as statics maintained for responses to both positive and negative training images. To compute the value each bin would receive, the classifier is run through a training set containing both positive and negative images. The total number of images and a sum of the image weights falling into each bin is maintained for positive and negative results. After the training set has been completely run, the mean image weight for the positive and negative results is computed for each bin. The value of that bin is then defined as the difference between the positive mean and the negative mean. Finally, a smoothing function is run over top of the bins to fill in any empty results.

The resulting classifier is evaluated to determine its error rate. The classifier with the lowest error rate is then selected to be added to the GentleBoost classifier and the training weights for each image are updated and normalized. The detection rate and false positive rate are also computed during the last stage.

Increases in performance were not observed as more classifiers were added to the system while testing the GentleBoost classifier. Those images that were originally misclassified stayed misclassified and did not receive additional weighting. The cause of this issue was the uniform distribution given to all training samples. A greater initial weight given to the positive training samples may have allowed algorithm to work as implemented. However, this approach was temporarily set aside in favor of the simpler AdaBoost algorithm (as described in Section IV.B) which could also be used to prove the validity of the data sets and in the future serve the same purpose in finding the best classifiers for each facial region.

F. Preliminary Training Set Verification Testing Results

An iterative testing strategy was used to verify the functionality of component objects and verify their correct functionality. An example of the output of such a process is shown and will be explained below.

All full facial region images are imported and 10% of these images are allocated to the evaluation set. The remaining 90% of the facial images are used as the training set. A negative set with the target size of 2,000 images is use to train each layer. A total of 170,800 Haar-like features are included in the feature pool to be used to create weak classifiers. Layers are added to the classifier until the overall false positive rate drops below 10^{-6}.

TABLE XX Result of AdaBoost classifier created to detect FaceR sub-regions.

Layer	Features	Layer DR	Layer FPR	Overall FPR
1	2	0.979592	0.108554	0.112316
2	5	1.000000	0.156578	0.018859
3	5	1.000000	0.418709	0.00826414
4	10	1.000000	0.119859	0.000918619
5	10	0.979592	0.110320	0.000110881
6	25	1.000000	0.000000	0.00000000

Upon adding the sixth layer to the classifier no additional false positives are added by the classifier, due to two factors. A maximum negative test value was set at 100,000 to limit the training time on limited computing power. This places an upper limit on the number of prospective false positives which can be examined to be added each step. Parallelization of the training algorithm is one possible solution to this issue. The second possible cause of the small resulting classifier size was the limited number of facial images used in training. Increasing the size of the training set would result in a more comprehensive detector.

IX. CONCLUSIONS AND FUTURE WORK

The study of facial movement and expression has been a prominent area of research since the early work of Charles Darwin. The Facial Action Coding System (FACS), developed by Paul Ekman, introduced the first universal method of coding and measuring facial movement. Facial expression recognition has been shown to be broken down into three distinctive subsections: Facial Feature Localization, Facial Action Recognition, and Facial Expression Classification. The first and most important stage in any facial expression analysis system as demonstrated to be the localization of key facial features. Localization must be accurate and efficient to ensure reliable tracking and leave time for computation and comparisons to learned facial models while maintaining real-time performance. Three specific research activities contributing to effective and real time facial expression recognition have been presented in this dissertation.

A preprocessing methodology to exempt the search area in low variance regions in an image frame for locating human faces has been developed. A top down quad-tree deconstruction of the frame was used to accomplish this task. Regional variance was computed and tested to determine if a given quadrant should be further broken down. If below a predefined threshold that region will be eliminated in a mask image. This procedure was continued until all sections were eliminated or a defined tree depth was reached. The resulting mask image was then smoothed and subjected to thresholding, merging the remaining valid search areas. The resulting filled mask was then used to determine whether a given sub-frame should be sent to a Viola-Jones face detection cascade. Results showed promise in reducing the number of sub-frames that must be considered for detection by 86.2% while maintaining a 100% face detection rate.

A real-time human computer interaction system which facilitates the automatic recognition of seven affective states viz. agreeing, concentrating, disagreeing, interested, thinking, unsure, and angry from a near infrared video stream has been developed. A Viola Jones face detector was trained to identify faces within the frame. The Active Appearance model was then used to place 23 landmark points around key areas of the eyes, eyebrows, and mouth. A prioritized binary decision tree then detected, based on the actions of these key points, if one of the seven emotional states occurs as frames pass. The completed system ran accurately and seamlessly on an Intel Pentium IV, 2.8 GHz processor with 512 MB of memory, and achieved a real-time frame rate of around 36 frames per second.

A method for training and operation, using an architectural configuration of a system of nested cascade detectors with the ability to focus in on a feature point of interest has also been developed. Each cascade was trained using positive and negative example sets of images resized to a smaller, 24×24 image patch to reduce the size of the possible feature set and decrease the training time. However, when operating on a prospective image frame the detector must be scaled and shifted to detect all possible sizes and positions of the object within the entire frame. The proposed architecture sought to perform a coarse-to-fine search. The entire object of interest was first identified followed by the identification of regions within the object and the specific features within those regions. The responses of the individual Haar-like features comprising those features at the coarse level were used to trigger the finer-level searches. In this manner each subsequent fine-level search could have been triggered as quickly and efficiently as possible. By taking this approach search area is reduced and false positives were reduced since finer level searches only occur in areas and scales where landmark features occurred over the training set.

Continuation of this research could start with the generation of complete detection cascades for all facial regions for which training sets have been generated. Upon the completion of an accurate tracking system, existing classification systems could be explored to form a real-time AU detection system. The study of predictive facial landmark tracking could be used to attempt to make the detector more efficient by reducing search time. Frame to frame temporal face movement can also be used to build a normalized model of the individual's facial geometry. To make an application capable of operating on an arbitrary individual it is necessary to start with a general model and refine this model until it closely mimics the geometry and possible movements of the unknown individual. A final extension would be to test the ability of the system to recognize specific characteristic facial motions and states. An emotion-based classification will be shown which utilizes various AU intensity combinations to detect mixed emotional states.

REFERENCES

[1] C. Darwin, *The Expression of the Emotions in Man and Animals, 3rd Edition*, P. Ekman, Ed. Oxford University Press, 2002.

[2] J. Cohn, Z. Ambadar, and P. Ekman, "Observer-based measurement of facial expression with the facial action coding system"," in *The handbook of emotion elicitation and assessment*, ser. Affective Science, J. A. C. . J. B. Allen, Ed. Oxford University Press, 2006.

[3] J. F. Cohn, "Foundations of human computing: facial expression and emotion," in *ICMI '06: Proceedings of the 8th international conference on Multimodal interfaces*. New York, NY, USA: ACM, 2006, pp. 233–238.

[4] P. Ekman and W. V. Friesen, *Unmasking the face: A guide to recognizing emotions from facial clues*. Oxford: Prentice-Hall, 1975.

[5] P. Ekman, W. V. Friesen, and J. C. Hager, *Facial Action Coding System*, A Human Face, Salt Lake City, UT, 2002, cD-ROM.

[6] M. Turk and M. Kölsch, "Perceptual interfaces," in *Emerging Topics in Computer Vision*, G. Medioni and S. Kang, Eds. Prentice Hall, 2004.

[7] N. O. Bernsen, *Multimodality in Language and Speech Systems*. Kluwer Academic Publishing, 2002, ch. Multimodality in language and speech systems. From theory to design support tool, pp. 93–148.

[8] A. Corradini, M. Mehta, N. O. Bernsen, and J. C. Martin, "Multimodal input fusion in human-computer interaction," in *Proceedings of the NATO-ASI Conference on Data*

Fusion for Situation Monitoring, Incident Detection, Alert and Response Management., Yerevan, Armenia, August 2003.

[9] G. Donato, M. S. Bartlett, J. C. Hager, P. Ekman, and T. J. Sejnowski, "Classifying facial actions," vol. 21, no. 10, pp. 974–989, Oct. 1999.

[10] M. Pantic and L. J. M. Rothkrantz, "Automatic analysis of facial expressions: the state of the art," vol. 22, no. 12, pp. 1424–1445, Dec. 2000.

[11] B. Fasel, J. Luettin, B. Fasel, and J. Luettin, "Automatic facial expression analysis: A survey," *Pattern Recognition*, vol. 36, pp. 259–275, 1999.

[12] S. Mitra and T. Acharya, "Gesture recognition: A survey," vol. 37, no. 3, pp. 311–324, May 2007.

[13] M. Beszedes and P. Culverhouse, "Comparison of human and automatic facial emotions and emotion intensity levels recognition," in *Proc. 5th International Symposium on Image and Signal Processing and Analysis ISPA 2007*, Sep. 27–29, 2007, pp. 429–434.

[14] S. Lucey, I. Matthews, C. Hu, Z. Ambadar, F. de la Torre, and J. Cohn, "Aam derived face representations for robust facial action recognition," in *Proc. 7th International Conference on Automatic Face and Gesture Recognition FGR 2006*, Apr. 2–6, 2006, pp. 155–160.

[15] H. van Kuilenburg, M. Wiering, and M. den Uyl, "A model based method for automatic facial expression recognition," 2005, pp. 194–205.

[16] L. Ma and K. Khorasani, "Facial expression recognition using constructive feedforward neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 34, no. 3, pp. 1588–1595, Jun. 2004.

[17] F. Dornaika and F. Davoine, "On appearance based face and facial action tracking," vol. 16, no. 9, pp. 1107–1124, Sep. 2006.

[18] I. Kotsia, N. Nikolaidis, and I. Pitas, "Facial expression recognition in videos using a novel multi-class support vector machines variant," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2007*, vol. 2, Apr. 15–20, 2007, pp. II–585–II–588.

[19] K. Anderson and P. W. McOwan, "A real-time automated system for the recognition of human facial expressions," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 36, no. 1, pp. 96–105, Feb. 2006.

[20] C.-S. Lee and A. M. Elgammal, "Facial expression analysis using nonlinear decomposable generative models," in *Proceedings of Analysis and Modelling of Faces and Gestures, Second International Workshop, AMFG 2005*, ser. Lecture Notes in Computer Science, W. Zhao, S. Gong, and X. Tang, Eds., vol. 3723, no. 2005. Beijing, China: Springer, Oct 2005, pp. 17–31.

[21] M. Pantic and L. J. M. Rothkrantz, "Expert system for automatic analysis of facial expressions," *Image Vision Comput.*, vol. 18, no. 11, pp. 881–905, 2000.

[22] ——, "Facial action recognition for facial expression analysis from static face images," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 34, no. 3, pp. 1449–1461, Jun. 2004.

[23] M. Pantic and I. Patras, "Dynamics of facial expression: recognition of facial actions and their temporal segments from face profile image sequences," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 36, no. 2, pp. 433–449, Apr. 2006.

[24] D. Vukadinovic and M. Pantic, "Fully automatic facial feature point detection using gabor feature based boosted classifiers," in *Proc. IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, Oct. 10–12, 2005, pp. 1692–1698.

[25] M. Valstar and M. Pantic, "Fully automatic facial action unit detection and temporal analysis," in *Proc. Conference on Computer Vision and Pattern Recognition Workshop*, Jun. 17–22, 2006, p. 149.

[26] M. S. Bartlett, G. Littlewort, M. Frank, C. Lainscsek, I. Fasel, and J. Movellan, "Fully automatic facial action recognition in spontaneous behavior," in *Proc. 7th International Conference on Automatic Face and Gesture Recognition FGR 2006*, Apr. 2–6, 2006, pp. 223–230.

[27] M. S. Bartlett, G. Littlewort, M. G. Frank, C. Lainscsek, I. R. Fasel, and J. R. Movellan, "Automatic recognition of facial actions in spontaneous expressions," *Journal of Multimedia*, vol. 1, no. 6, pp. 22–35, 2006.

[28] G. Littlewort, M. S. Bartlett, I. R. Fasel, J. Susskind, and J. R. Movellan, "Dynamics of facial expression extracted automatically from video," *Image Vision Comput.*, vol. 24, no. 6, pp. 615–625, 2006.

[29] Y. Wang, H. Ai, B. Wu, and C. Huang, "Real time facial expression recognition with adaboost," in *Proc. 17th International Conference on Pattern Recognition ICPR 2004*, vol. 3, Aug. 23–26, 2004, pp. 926–929.

[30] P. S. Aleksic and A. K. Katsaggelos, "Automatic facial expression recognition using facial animation parameters and multistream hmms," vol. 1, no. 1, pp. 3–11, Mar. 2006.

[31] Y. Zhang and Q. Ji, "Active and dynamic information fusion for facial expression understanding from image sequences," vol. 27, no. 5, pp. 699–714, May 2005.

[32] R. A. Kaliouby, "Mind-reading machines: automated inference of complex mental states," University of Cambridge Computer Laboratory, Tech. Rep. 636, July 2005.

[33] R. El Kaliouby and P. Robinson, "Mind reading machines: automated inference of cognitive mental states from video," in *Proc. IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, 2004, pp. 682–688.

[34] R. Kaliouby and P. Robinson, "Generalization of a vision-based computational model of mind-reading." Springer, 2005, pp. 582–589.

[35] C. Shan, S. Gong, and P. W. McOwan, "Recognizing facial expressions at low resolution," in *Proc. IEEE Conference on Advanced Video and Signal Based Surveillance AVSS 2005*, Sep. 15–16, 2005, pp. 330–335.

[36] X. Feng, J. Cui, M. Pietikäinen, and A. Hadid, "Real time facial expression recognition using local binary patterns and linear programming," in *MICAI 2005: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, . H. T. A. Gelbukh, A. de Albemoz, Ed. Springer Berlin / Heidelberg, 2005, vol. 3789/2005, pp. 328–336.

[37] D. Cristinacce and T. Cootes, "Facial feature detection using adaboost with shape constraints," in 14th *British Machine Vision Conference, Norwich, England*, 2003, pp. 231–240.

[38] D. Cristinacce and T. F. Cootes, "Facial feature detection and tracking with automatic template selection," in *Proc. 7th International Conference on Automatic Face and Gesture Recognition FGR 2006*, Apr. 2–6, 2006, pp. 429–434.

[39] D. Cristinacce and T. Cootes, "Feature detection and tracking with constrained local models," in 17th *British Machine Vision Conference, Edinburgh, UK*, 2006, pp. 929–938.

[40] P. F. Felzenszwalb and D. P. Huttenlocher, “Pictorial structures for object recognition,” *International Journal of Computer Vision*, vol. 61, p. 2005, 2003.

[41] L. Wang, B. Zou, and J. Sun, “Facial features location by analytic boosted cascade detector,” in *Computational Intelligence and Security*, ser. Lecture Notes in Computer Science, Y. H. et al., Ed. Springer Berlin / Heidelberg, 2005, vol. 3802, pp. 959–964.

[42] I. Cohen, A. Garg, and T. S. Huang, “Emotion recognition from facial expressions using multilevel hmm,” in *in In Neural Information Processing Systems*, 2000.

[43] N. Esau, E. Wetzel, L. Kleinjohann, and B. Kleinjohann, “Real-time facial expression recognition using a fuzzy emotion model,” in *Proc. IEEE Int. Fuzzy Systems Conf. FUZZ-IEEE 2007*, 2007, pp. 1–6.

[44] A. Dhall, A. Asthana, R. Goecke, and T. Gedeon, “Emotion recognition using phog and lpq features,” in *Proc. IEEE Int Automatic Face & Gesture Recognition and Workshops (FG 2011) Conf*, 2011, pp. 878–883.

[45] C. Busso, Z. Deng, S. Yildirim, M. Bulut, C. M. Lee, A. Kazemzadeh, S. Lee, U. Neumann, and S. Narayanan, “Analysis of emotion recognition using facial expressions, speech and multimodal information,” in *Sixth International Conference on Multimodal Interfaces ICMI 2004*. ACM Press, 2004, pp. 205–211.

[46] M. Meghjani, F. Ferrie, and G. Dudek, “Bimodal information analysis for emotion recognition,” in *Proc. Workshop Applications of Computer Vision (WACV)*, 2009, pp. 1–6.

[47] D. Datcu and L. J. M. Rothkrantz, “Emotion recognition using bimodal data fusion,” in *Proceedings of the 12th International Conference on Computer Systems and*

Technologies, ser. CompSysTech '11. New York, NY, USA: ACM, 2011, pp. 122–128. [Online]. Available: http://doi.acm.org/10.1145/2023607.2023629

[48] T. F. Cootes, G. Edwards, and C. Taylor, "A comparative evaluation of active appearance model algorithms," in *9th British Machine Vision Conference*. BMVA Press, 1998, pp. 680–689.

[49] M. J. Jones and P. A. Viola, "Fast mulit-view face detection," Mitsubishi Electronic Research Laboritories, Cambridge, MA, Tech. Rep. TR2003-96, July 2003.

[50] P. A. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, May 2004.

[51] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Scienes*, vol. 2, no. 1, pp. 680–689, 1997.

[52] I. Fasel, B. Fortenberry, and J. Movellan, "A generative framework for real time object detection and classification," *Computer Vision and Image Understanding*, vol. 98, no. 1, pp. 182–201, April 2005.

[53] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *Proceedings of the 2002 International Conference on Image Processing*, vol. 1. IEEE, September 2002, pp. 900–903.

[54] B. Wu, H. Ai, C. Huang, and S. Lao, "Fast rotation invariant multi-view face detection based on real adaboost," in *Proc. Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, May 17–19, 2004, pp. 79–84.

[55] B. Silverman, *Density Estimation for Statistics and Data Analysis*. London: Chapman and Hall, 1986.

[56] C.-W. Park and M. Park, "Fast template-based face detection algorithm using quad-tree template," *Journal of Applied Sciences*, vol. 6, pp. 795–799, 2006. [Online]. Available: http://scialert.net/abstract/?doi=jas.2006.795.799

[57] G. J. Sullivan and R. L. Baker, "Efficient quadtree coding of images and video," vol. 3, no. 3, pp. 327–331, 1994.

[58] D. Saupe and S. Jacob, "Variance-based quadtrees in fractal image compression," *Electronics Letters*, vol. 33, no. 1, pp. 46–48, 1997.

[59] P. Barroso, J. Amaral, A. Mora, J. Fonseca, and A. Steiger-Garção, "A quadtree based vehicules recognition system," in *4th International Conference on Optics, Photonics, Lasers and Imaging (ICOPLI 2004)*, 14-16 January 2004.

[60] L. M. Kaplan, S.-M. Oh, and J. H. McClellan, "Detection of broadside targets during image formation using a quadtree approach," in *Proc. Record of the IEEE 2000 Int. Radar Conf*, 2000, pp. 104–109.

[61] P. Ekman, *The Argument and Evidence about Universals in Facial Expressions of Emotion*. New York: Wiley, 1989.

[62] P. Ekman and W. Friesen, *The Facial Action Coding System: A Technique for the Measurement of Facial Movement*. San Francisco: Consulting Psychologists Press, 1978.

[63] K. R. Scherer and P. Ekman, *Handbook of methods in nonverbal behavior research*. Cambridge and New York: Cambridge University Press, 1982.

[64] M. Suwa, N. Sugie, and K. Fujimora, "A preliminary note on pattern recognition of human emotional expression," *International Joint Conference of Pattern Recognition*, pp. 408–410, 1978.

[65] Y.-I. Tian, T. Kanade, and J. F. Cohn, "Recognizing action units for facial expression analysis," vol. 23, no. 2, pp. 97–115, Feb. 2001.

[66] S. Baron-Cohen, O. Golan, S. Wheelwright, and J. J. Hill, *Mind Reading: The Interactive Guide to Emotions*. Jessica Kingsley Publishers, 2004.

[67] ——, "A new taxonomy of human emotions," 2004.

[68] (2002) Facetracker, facial feature tracking sdk. Neven Vision.

[69] V. Bruce, *Recognizing Faces*. East Sussex, UK: Lawrence Erlbaum, 1986.

[70] T. Kanade, J. F. Cohn, and Y. Tian, "Comprehensive database for facial expression analysis," in *Proc. Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, Mar. 28–30, 2000, pp. 46–53.

[71] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," Caltech, Tech. Rep. 7694, 2007, http://authors.library.caltech.edu/7694/.

VITA

Adam Livingston studied Electrical and Computer Engineering at Old Dominion University (ODU), Norfolk, Virginia. Adam received a Bachelor of Science degree in Computer Engineering with a minor in Electrical Engineering from ODU in 2004, a Masters degree in Computer Engineering from ODU in 2006, and Doctorate in Electrical and Computer Engineering in May, 2012.

Educational Background

PhD,	Electrical and Computer Engineering, Old Dominion University, May 2012.
Masters,	Computer Engineering, Old Dominion University, May 2006.
Bachelors,	Computer Engineering, Old Dominion University, May 2004.

Department of Study

Department of Electrical and Computer Engineering, 231, Kaufman Hall, Norfolk,VA, 23529.

Research Laboratory

Computational Intelligence and Machine Vision Laboratory, Suite 202, 4111, Monarch Way, Norfolk, VA, 23508.

Selected Publications

1. Adam Livingston, Ming-Jung Seow, and K. Vijayan Asari, "A real-time emotion detection system for human computer interaction: A binary decision tree approach," ***Recent Advances in Control Systems, Robotics and Automation - Third Edition****, ISBN 978-88-901928-7-6*, vol. 2, pp. 151-159, January 2009.

2. Adam Livingston and K. Vijayan Asari, "Regional variance dependant sub-frame reduction for face detection in video streams," ***IEEE Computer Society Proceedings of the International Workshop on Applied Imagery and Pattern Recognition - AIPR 2007***, Washington, D.C., pp. 89-94, October 10-12, 2007.

3. Adam Livingston, Hau T. Ngo, Ming Zhang, Li Tao, and K. Vijayan Asari, "Design of a real time system for nonlinear enhancement of video streams by an integrated neighborhood dependent approach," ***IEEE Computer Society Proceedings of the International Symposium on VLSI – ISVLSI 2005***, Tampa, Florida, pp. 301-302, May 11 – 12, 2005.