Head Pose Estimation and its Application in TV Viewers' Behavior Analysis

by

Siyu Wu

B.Eng., University of Science and Technology of China, 2012

Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Applied Science

in the
School of Engineering Science
Faculty of Applied Sciences

Approval

Name:	Siyu Wu
Degree:	Master of Applied Science
Title:	*Head Pose Estimation and its Application in TV Viewers' Behavior Analysis*
Examining Committee:	**Dr. Ivan V. Bajić** (chair) Associate Professor

Dr. Jie Liang
Senior Supervisor
Professor

Dr. Mirza Faisal Beg
Supervisor
Professor

Dr. Shahram Payandeh
External Examiner
Professor
School of Engineering Science
Simon Fraser University

Date Defended:	4 December 2015

Abstract

As a reliable indicator of visual gaze direction, head pose implies a person's visual attention and interest. Therefore, head pose information extracted from face images serves as important input in many applications. In this thesis, a coarse-to-fine head pose estimation method is proposed, by decomposing the original pose space in a hierarchical structure. The estimation begins with a coarse step to identify a subspace that encompasses a set of head pose candidates. Then a subsequent fine estimation is conducted within the subspace, generating a refined result. Besides, to eliminate irrelevant information within a face image, we propose to detect Region of Interest (ROI) by exploring importance degree of image points. Furthermore, we build an application of analyzing TV viewers' behaviors from video recordings, by integrating face detection, face tracking and head pose estimation. Based on head pose and face motion, a viewer's behavior is identified to be focused or unfocused.

Keywords: Head pose; coarse-to-fine estimation; hierarchical structure; region of interest; TV viewers' behaviors

Acknowledgements

First of all, I would like to give my sincere thanks to my senior supervisor Dr. Jie Liang for his support, guidance and patience in the past three years. He helps me overcome difficulties in my research work and achieves improvement in reasoning and critical thinking, which are very important in my future career.

I would also like to warmly thank Dr. Mirza Faisal Beg for being my committee member and offering me valuable suggestions to improve the thesis. Thanks to Dr. Ivan V. Bajić for chairing the presentation of my thesis and Dr. Shahram Payandeh for being the examiner. Special thanks to Dr. Jason Ho for cooperating with us in this project. He offers me great help in collecting data and recording videos.

I also want to express my thanks to instructors who ever taught me in SFU. Thanks to Dr. Ivan V. Bajic, who taught me the basic knowledge of stochastic systems. Thanks to Dr. Oliver Schulte, who guided me to learn machine learning techniques. And thanks to Dr. Rodney G. Vaughan, who led me into the area of statistical signal processing. Thanks to Dr. Mirza Faisal Beg, who introduced novel ideas in linear systems to me.

I would like to express my appreciation to my labmates for their help during my studies. It is a wonderful experience working with them.

Last but not least, I want to express my foremost gratitude to my parents, for their love all through my life.

Table of Contents

List of Tables

List of Figures

List of Acronyms

AAM	Active Appearance Model
AdaBoost	Adaptive Boosting
ASM	Active Shape Model
BEM	Block Energy Map
CED	Cumulative Error Distribution
FPS	frames per second
GPR	Gaussian Process Regression
HOG	Histogram of Oriented Gradients
KLT	Kanade–Lucas–Tomasi
k-NN	k-nearest neighbors
LARR	Locally Adjusted Robust Regressor
LBP	Local Binary Pattern
MAE	Mean Absolute Error
RBF	Radial Basis Function
RGB	Red-Green-Blue
ROI	Region of Interest
SDA	Subclass Discriminant Analysis
SIFT	Scale-Invariant Feature Transform
SVM	Support Vector Machine
SVR	Support Vector Regression

VFOA	Visual Focus of Attention
2D	Two-Dimensional
3D	Three-Dimensional

Chapter 1

Introduction

With the rapid development of computer technologies and an exploding growth of digital images, computer vision continues to be one of the most active fields of research. In this field, human face images gain great interest. The fundamental face related topic is face detection which aims to identify face locations from digital images. Besides, there is lots of work devoted to further extract high-level information from detected faces. One important branch is our focus in this thesis—head pose estimation.

On top of the question "where is the person" concerned in face detection, head pose estimation addresses a new question—"where does the person look?", based on the fundamental assumption that head pose is highly correlated with the direction of visual gaze [62]. Determining the direction of the visual field, a person's gaze direction is reflected from his eyes. However, directly estimating gaze direction from narrow eye regions in a face image suffers from the sheer difficulty of eye detection.

Fortunately, the tough problem can be solved in a different way. Physiological investigations reveal that gaze direction is indicated by the orientation of a human head, i.e., head pose [38]. Head pose can be estimated from the face image, which contains much richer information than merely eye regions. Consequently, estimating a person's head pose provides a reliable approximation of his visual gaze.

Given a face image captured under the view of a camera, head pose estimation is the process of figuring out the person's head orientation relative to the camera; in a more rigorous sense, the orientation is strictly relative to a real world coordinate system, which requires the knowledge of camera parameters [49]. However, in most cases, camera calibration is undesired, and the orientation strictly in a camera view is needed. Therefore, head pose relative to the camera is our concern in this thesis.

Head pose can be viewed as a type of body language, providing nonverbal communication cues. Besides, the directional information conveyed by the head pose encapsulates multiple meanings. According to [60], the direction reflects a person's Visual Focus of Attention

(VFOA), implying his interest with respect to a visual target and reaction to a visual stimuli.

Based on the aforementioned findings, the most obvious application of head pose estimation is building a hands-free interface. For instance, knowledge of a person's head pose may directly control a device designed for disabled people, or act as a replacement of the mouse in human-computer interaction [45]. Head pose estimation can also be applied in human behavior understanding, such as monitoring drivers' attention and analyzing inclination of passers-by to an outdoor advertisement [48, 60]. Another possible application is social behavior modeling. Apart from verbal communication, analyzing head pose of a group of people is a good way of understanding interaction between them [1]. Furthermore, head pose can be used to understand the real world in augmented reality [4].

In aforementioned applications, videos are the most commonly used inputs, which are composed of a number of consecutive frames. To extract head pose information over time, such an application is built on an integration of the following tasks:

1. Face detection: to detect face locations
2. Face tracking: to track faces in the subsequent frames
3. Head pose estimation: to determine head pose from faces using head pose estimation method
4. Head pose analysis: to interpret head pose information

Among them, the first three tasks create a head pose estimation system, aimed at extracting head pose information from video recordings.

The first step in building an application is face detection, which is also a prerequisite for estimating head pose on static images. Introduced by face detection, one inherent difficulty for head pose estimation is that the detected faces would contain unnecessary and even distractive information in terms of head pose, the most common of which is the background information in the marginal area and hair appearing on the forehead. Other distraction might be accessories such as eyeglasses. Another major difficulty for estimating head pose comes from variations irrelevant to head pose among individuals, such as facial expression, skin color and appearance. Besides, images may be captured under different lighting conditions subject to various environments. Also a common issue in computer vision field—image noise remains a problem in head pose estimation.

Considering aforementioned difficulties, head pose estimation is a challenging and perplexing task. Our work is motivated by the urge to improve the performance of head pose estimation in order to enhance applications based on head pose. Particularly, we are interested in an application in which head pose is used to analyze TV viewers' behaviors. A TV viewer's head pose relative to the TV screen implies whether the person is watching TV or not. Therefore analyzing a viewer's head pose gives a deep understanding about how engaged the person is with the TV programme.

1.1 Contribution

This thesis provides a detailed study of estimating head pose. We describe existing head pose estimation methods and head pose estimation systems. We then propose a coarse-to-fine head pose estimation method. We also conduct an experiment, and apply the proposed head pose estimation method to analyze TV viewers' behaviors. The result is very helpful in marketing research.

A summary of our contributions is given as follows.

Firstly, we divide the pose space which contains all the pose configurations into a number of subspaces and thereby decompose the original pose space in a hierarchical structure.

Secondly, based on the hierarchical pose space structure, we propose a two-step estimation mechanism—coarse-to-fine estimation. Unlike traditional methods that directly conduct estimation in the original pose space, our method begins with a coarse estimation which identifies a unique subspace that encompasses a set of candidate head pose configurations. The subsequent step makes a refined estimation in the subspace and generates the final estimation result.

Thirdly, to eliminate irrelevant information within a face image, we propose a ROI detection module by exploring importance degree of different image points and iteratively selecting the most important ones.

Fourthly, we use the proposed method to analyze TV viewers' behaviors. This kind of analyses can provide useful information for some applications such as marketing research. For example, it can be used to study the effectiveness of ads in TV broadcasting. To get stable face locations in a sequence of frames, face tracking is based on weighted Kanade–Lucas–Tomasi (KLT) algorithm. Besides, to eliminate invalid tracking results caused by occlusion, we propose a consistency checking module. Furthermore, by selecting a base frame, a TV viewer's behavior is determined from the relative head pose change and face motion.

1.2 Thesis Structure

The rest of this thesis is organized as follows. Chapter 2 describes the head pose basics, and then provides an overview of prior work in head pose estimation area and related techniques in head pose estimation method. Our proposed coarse-to-fine head pose estimation method is presented in Chapter 3, including general ideas and details about hierarchical representation of the pose space, coarse-to-fine estimation, and detection of ROI. In Chapter 4, we elaborate on analyzing TV viewers' behaviors from video recordings. Chapter 5 presents performance of coarse-to-fine estimation method. Additionally, performance of TV viewers' behavior analysis is given in this chapter. Finally, Chapter 6 concludes the thesis.

Chapter 2

Background

In this chapter, we first describe basics of head pose (Chapter 2.1). Then we give an overview of previous research on head pose estimation methods. We further divide existing methods into three categories, followed by an introduction of relevant work and analysis of the performance of methods in each category (Chapter 2.2). After that, we give an overview of prior studies on head pose estimation systems (Chapter 2.3). Finally we present preliminaries of our proposed head pose estimation method (Chapter 2.4).

2.1 Head Pose Basics

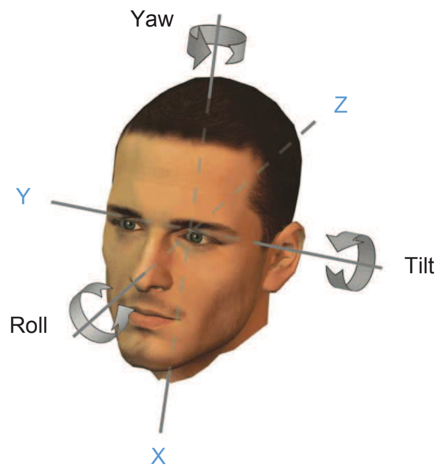
Generally speaking, head pose denotes the orientation of a person's head. In computer vision field, head pose can be interpreted in various ways. Semantically, head pose may be coarsely characterized by a few discrete orientations, such as frontal view, half profile and profile [70]. However, to better depict the orientation of a head, researchers are more interested in quantitative result, thus angular measurement is usually applied to describe the head pose [49].

Given a 3D face model shown in Figure 2.1a, head pose is measured by Euler angles in three degrees of freedom, specified by:

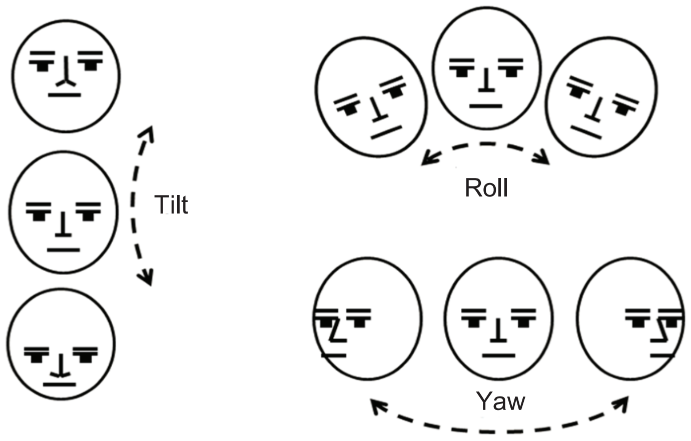
- Yaw: generated from rotation along x axis
- Tilt: generated from rotation along y axis
- Roll: generated from rotation along z axis

Figure 2.1a shows the three angles correspond to rotations with respect to three orthogonal axes. By projecting the 3D model onto a 2D plane, simulated 2D face images generated from three kinds of rotations are illustrated in Figure 2.1b. From Figure 2.1b, roll corresponds to rotation within the 2D plane defined by the face image, while yaw and tilt both correspond to rotations out of that plane. Compared to in-plane rotation, out-of-plane rotation is more difficult to estimate due to the nonlinear image transformation.

In general, movement of a human head can be decomposed into the three rotations. Under normal circumstances, the in-plane head rotation is insignificant, while out-of-plane head rotations take place for most of the time. As a consequence, yaw and tilt are the two most important elements in determining a person's head pose. In this thesis, regardless of minor change in roll angle, head pose is described in terms of yaw and tilt. Therefore, by figuring out yaw and tilt, a person's head pose can be identified.



(a)



(b)

Figure 2.1: (a) Three angles in the 3D coordinate system; (b) 2D Face images generated by three kinds of rotations.

2.2 Overview of Head Pose Estimation Methods

The goal of head pose estimation is to determine head pose from face images. Generally speaking, a head pose estimation method is supposed to have the following properties [49]:

- It is person-independent.
- It is effective and robust.
- It runs fast in order to enhance real-time applications.

Learning a head pose estimation method usually relies on a head pose image database as the training set. A head pose image database consists of face images for different people with all possible pose configurations in the designated pose space, and corresponding ground truth denoting yaw and tilt in degrees. Besides, for a head pose estimation method, its related database should follow its requirement for the format of input face images.

The reason why there are various image formats with respond to different methods lies in the fact that each head pose estimation method implicitly depends on a specific hardware layout. For instance, images used in some work are acquired by stereo cameras, in order to describe 3D information of facial parts. Examples of stereo image based methods can be found in [21, 36], where a Kinect sensor is employed to obtain the depth data. Also Seemann et al. [57] claimed that depth information acquired by MEGA-D stereo camera improves estimation accuracy and robustness of the system.

Some other existing methods [5, 47, 55] adopt images captured by several cameras at different locations as inputs and estimate head pose by fusing features of these images. According to [55], a delicate multi-camera system covers a much larger space than a single camera. In addition, cross-camera correspondence makes head pose estimation reliable and robust [5].

Although methods relying on images obtained by a stereo camera or multiple cameras can achieve excellent performance, accessibility and feasibility of the hardware setup must be taken into account. In general, it is a perplexing task to build a multi-camera system. Besides, despite the fact that stereo cameras are becoming increasingly available, people show more inclination to convenient and portable devices, such as smart phones. In addition, many applications based on head pose are designed for surveillance purpose, using videos recorded by a regular RGB camera. Therefore, we are more interested in head pose estimation methods with monocular 2D images as inputs. Also, to guarantee a wide range of applications, we prefer required resolution of images to be moderate.

Existing head pose estimation methods with our designated setting can be divided into several categories. Based on the machine learning technique employed in a method, there are classification based, regression based, and hybrid methods [29]. The main difference lies in determining the pose space to be discrete or continuous. Usually, classification based methods achieve higher accuracy, while regression based methods generate error within a smaller range. Based on the strategy on how to deal with face images, current head pose estimation methods can be mainly divided into the following three categories: geometric based methods, model based methods, and appearance based methods [69].

2.2.1 Geometric Based Methods

Geometric based methods depend on interpreting facial geometry. For a face image, by locating a set of facial features, such as eyes, nose and mouth, head pose is inferred from the positional information of facial features (Figure 2.2a).

For example, head pose estimation methods proposed in [56, 64] are built with the help of face and facial feature detection method developed by Viola and Jones [65]. Based on detected nose, eyes and mouth parts within a face image, head pose is determined from the geometric configuration. In a more complicated way, a regression model is trained in [11] in order to precisely locate feature points. After that, another regression model is trained to estimate head pose from the feature layout (Figure 2.2b).

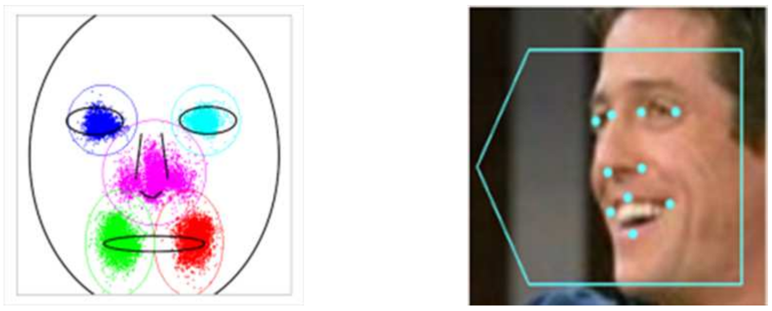

(a) (b)

Figure 2.2: (a) Facial features (shown in color regions) in head pose estimation task [64]; (b) Head pose estimation based on geometric layout of facial (blue) points [11].

Geometric based methods put special requirement on the training set. That is, for images in the training set, ground truth not only consists of corresponding yaw and tilt angles, but also contains coordinates of designated facial features. Such coordinates for training images are usually collected manually, to avoid error introduced in facial feature detection.

Performance of geometric based methods is affected by the feature detector to a large degree. In this sense, absolute visibility of required facial features is a basic requirement. As a result, occlusion caused by hair or accessories (e.g., glasses) may give rise to huge error in head pose estimation result. Moreover, resolution of the inputs directly affects the performance of feature detector and geometric based methods. For low-resolution images with the lack of clear details, geometric based methods show little possibility of achieving good performance.

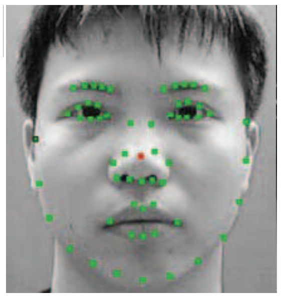
2.2.2 Model Based Methods

The basis of model based methods is building distinctive models corresponding to face images under different poses. Model based methods can be viewed as template matching methods: by fitting models to a face image, the best match could be found; the matched model is then used to describe the face image and determine the head pose.

Active Shape Model (ASM) [9] and Active Appearance Model (AAM) [8] are two representative models, widely used in analyzing face images. ASM is formed by facial feature points. The feature points not only include pixels around facial organs, but also consist of edge pixels depicting the face contour. Then a shape model is generated by concatenating the points as a whole. AAM is evolved from ASM. Besides the use of shape constraint exploited in ASM, AAM also takes the advantage of texture information of an image. In [34], given a face image, totally 58 feature points are extracted by pre-learned ASMs (Figure 2.3a). In [33], Ishikawa demonstrated the effectiveness of AAM. Based on extracted feature points, a feature vector is generated. After that, by learning the mapping from feature space to pose space, head pose for the face image is estimated.

In addition to ASM/AAM based methods, there exists other work devoted to building a more effective face model. In [71], several face models are trained by considering face and landmark detection along with pose estimation as a unified problem. The unified models prove to be more accurate than AAM.

Unfortunately, similar to geometric based methods, model based methods are limited by their requirements for the size and resolution of input face images. Such methods perform well for face images with high resolution, whereas for low-resolution images, most probably these methods would not work. Additionally, it usually takes a long time to run a model based method, which can be a restriction for real-time applications.


(a)

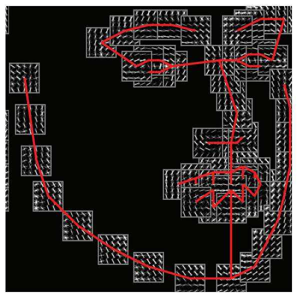

(b)

Figure 2.3: (a) Facial feature (green) points detected by ASMs [34]; (b) One of the unified face models built in [71].

2.2.3 Appearance Based Methods

Unlike methods of the aforementioned two categories, appearance based methods extract none of the isolated facial parts. On the contrary, appearance based head pose estimation is carried out by extracting features from the entire face image.

Consequently, extracting an effective feature vector to represent the face image is a critical step. The feature vector is also denoted as image descriptor. A feature vector in head pose estimation task is supposed to emphasize pose variation, and suppress other variations irrelevant to pose, e.g., identity, expression, lighting [41]. Existing methods of this category propose the use of many different image features. The simplest one is normalized pixel value, used in [29, 42]; gradient based features, such as localized gradient orientation [48], histogram of oriented gradients [18, 68] and covariance of oriented gradients [17], are considered to be effective in emphasizing facial contours; there are more complicated features, such as Scale-Invariant Feature Transform (SIFT) features [30] and Gabor feature [41, 51].

In appearance based methods, commonly used machine learning algorithms include k-nearest neighbors (k-NN) [17, 69, 41], support vector machine [48, 29], neural network [66, 61] and random forest [40, 35].

With a discriminative image descriptor and an effective machine learning technique, appearance based methods show great advantages in the following aspects: (1) by treating an image as a whole, appearance based methods can avoid the tough tasks of facial point locating and modeling; (2) appearance based methods work well even for low-resolution images; (3) appearance based methods run fast and can be applied to real-time applications.

2.3 Overview of Head Pose Estimation Systems

As stated in Chapter 1, a head pose estimation system is designed to extract head pose information from video recordings. Face detection, face tracking and head pose estimation are three main components in a head pose estimation system. In this section, previous work in building a head pose estimation system is reviewed from the three aspects.

To initialize a system, face detection is firstly conducted to detect all the possible face locations from a given frame. In [6], face detection relies on human detection results, i.e., a person's head is located from the detected human body. In this system, people are under the surveillance of an overhead camera, and face images are of small size. In this sense, directly conducting face detection is difficult, while human detection provides reliable face locations. In other work [32, 54], a camera is aimed to capture people's head movements, therefore face detection is directly conducted on captured images. Face detection methods used in these systems include Viola-Jones method [65], skin color based method, and elliptical edge detection method [3]. Making use of the elliptical shape of faces, elliptical edge detection detects face contour from edge pixels.

Face tracking is a key component in a continuous head pose estimation system, with the purpose of estimating instantaneous face locations. Huang et al. [32] used a Kalman filter to track faces under the assumption of constant velocity, with the upper-left corner coordinates and the size of the face as state vectors. Ren et al. [54] exploited color information to facilitate face tracking by searching the neighborhood area of previous face location. Ishikawa [33] explored the use of AAM model to track a face along with its feature points. To cope with the occlusion problem in multi-person tracking, Smith et al. [60] developed a Bayesian network to jointly track a person's body and face.

Based on face tracking results, head pose estimation is carried out on frames containing faces, by utilizing the developed estimation method. Appearance based method is adopted in [54], assisted by k-NN in the feature space. In [2], head pose is determined by randomised ferns built with Histogram of Oriented Gradients (HOG) and color based image feature. In [32], a hidden Markov model is developed to estimate head pose.

2.4 Preliminaries

Considering pros and cons of aforementioned three categories of methods, we propose an appearance based head pose estimation method. In addition, in order to achieve high accuracy, classification based machine learning techniques are used in our proposed method. In the following part, we discuss candidate face detection algorithms. Then we present image descriptors and machine learning techniques employed in our proposed head pose estimation method.

2.4.1 Face detection

Head pose information is extracted from face images. Therefore, face detection is a prerequisite for head pose estimation. The most commonly used face detection algorithm is developed by Paul Viola and Michael Jones [65], thus denoted as Viola-Jones algorithm. To detect whether a human face exists or not, face detection can be viewed as a binary classification task. In Viola-Jones algorithm, images are represented by Haar-like features (Appendix A.1), in which calculating the sum of pixels within rectangle areas is universally involved, hence integral image representation (Appendix A.2) is introduced to reduce the computational cost. Based on Haar-like image features, Adaptive Boosting (AdaBoost) (Appendix A.3) is employed to build a number of classifiers. Finally, the classifiers are combined sequentially and form a cascade detector.

Another typical method for finding face locations is based on skin pixel detection, making use of color difference between skin and non-skin pixels [7]. From a huge set containing amounts of pixels of two types, two RGB histograms are created to model the color distributions. Denoting the skin-pixel histogram as H and the non-skin one as h, the skin likelihood for a pixel with intensity (r, g, b) is computed by $\log(H(r, g, b)/h(r, g, b))$. By setting a threshold of the skin likelihood, skin pixels in an image can be detected. On top of the skin detection performed on individual pixels, face locations are detected by connected-component labeling (Appendix B).

In general, Viola-Jones algorithm can achieve reliable detection results in terms of faces within a certain pose range. However, for faces under poses out of the range, Viola-Jones algorithm would most probably fail. In comparison, the color-based skin detection method can avoid the suffering from pose variation and occlusion, while its disadvantage comes from the sensitivity to lighting conditions. Also, it only applies to color images.

2.4.2 Image Descriptors

Before presenting details of candidate image descriptors, it is noteworthy that there exists a basic understanding that a person's head pose is closely associated with geometry characteristics of the face image [17], which explains why there are plentiful geometric based head pose estimation methods. For appearance based methods, the name stems from the fact that the whole image rather than isolated facial part acts as input. From our knowledge, appearance based methods still rely on extracting features that emphasize geometry structure of a face image. For example, prior work [17, 48, 68] has proved that gradient based features reflect directional change in pixel intensity.

Gabor Feature

Generated from Gabor filters which are designed in terms of orientation and location, Gabor feature is believed to convey more orientational and spatial information than gradients. In addition, Daugman [13] claimed that image interpretation in mammalian visual cortex can be explained by Gabor function. Consequently, Gabor based image representation is speculated to accord with human visual understanding [44].

Gabor Filter Gabor feature is based on a set of 2D Gabor filters [12], which can be expressed by

$$\varphi_{\Pi(f_u,\theta_v,m,n)}(x,y) = \frac{f_u^2}{\pi\gamma\eta}\exp(-(\frac{f_u^2}{\gamma^2}x_r^2 + \frac{f_u^2}{\eta^2}y_r^2))\exp(j2\pi f_u x_r) \tag{2.1}$$

$$x_r = (x - x_c)\cos\theta_v + (y - y_c)\sin\theta_v \tag{2.2}$$

$$y_r = -(x - x_c)\sin\theta_v + (y - y_c)\cos\theta_v \tag{2.3}$$

$$f_u = \frac{f_{\max}}{\sqrt{2^u}},\ \theta_v = \frac{v}{V}\pi \tag{2.4}$$

where $x \in [x_c - \frac{m+1}{2}, x_c + \frac{m+1}{2}]$, and $x \in N^*$; $y \in [y_c - \frac{n+1}{2}, y_c + \frac{n+1}{2}]$, and $y \in N^*$; $u = 0, ..., U-1$, and $v = 0, ..., V-1$.

In the above formulas, $\exp(-(\frac{f_u^2}{\gamma^2}x_r^2 + \frac{f_u^2}{\eta^2}y_r^2))$ is a 2D elliptical Gaussian; $\exp(j2\pi f_u x_r)$ is a plane wave with frequency f_u, which is usually viewed as modulation for the Gaussian part; (x_c, y_c) is the center of the filter; θ_v is the counterclockwise rotation of the filter, intuitively expressed by the angle between Gaussian's major axis and the horizontal line; (x_r, y_r) is a pair of spatial coordinates after rotation; $f_{\max}$ denotes the highest frequency; m and n denote the size of Gabor filter along two dimensions; π, γ, η are all constants; $j = \sqrt{-1}$.

Equations 2.1 to 2.3 indicate that a 2D Gabor filter is designed with several parameters—orientation, spatial frequency and spatial coordinates. With adjustable parameters in Equation 2.4, a filter bank, i.e., a Gabor filter array of size $U \times V$, is generated (U and V are related to the number of filters), and each filter has its own orientational and frequency emphasis. Denis Gabor [25] proved that a 2D Gabor filter is an optimal solution to preserve spatial as well as frequency information. Another remarkable point is that a Gabor filter is defined by a complex function. Researchers are usually interested in its real part and magnitude.

Gabor Image For a 2D image, its Gabor image is the convolution result with a 2D Gabor filter, given by

$$G(x, y) = I(x, y) * \varphi(x, y) \tag{2.5}$$

where I is the input image, usually in grayscale format; φ is a Gabor filter from Equation 2.1; G is the output Gabor image. In this thesis, $*$ denotes convolution operation. Since the filter φ is defined by a complex function, convolution result G is also complex, with the same size as image I.

A Gabor image G reveals the features of image I under a specific orientation and frequency. Therefore, by means of a Gabor filter bank, an array of Gabor images provides an extensive measurement of the original image and extracts information from different perspectives.

Gabor Feature To generate a representation of an image, an image descriptor is formed by Gabor image, denoted as Gabor feature.

The procedure of generating Gabor feature is described below. Since Gabor images encapsulate orientational and spatial information, there is no need to further process them. From a Gabor image G, a 1D column vector v is generated by reshaping its magnitude, i.e., $|G|$. Considering uniqueness of each Gabor image in an image array, Gabor feature f is a concatenation of the individual column vectors generated by all the Gabor images.

With an original image I of size $W \times H$ and a Gabor image array of size $U \times V$, Gabor feature is of dimension $U \cdot V \cdot W \cdot H$. Considering curse of dimensionality for most machine learning techniques, Gabor images are downsampled to size $W_d \times H_d$. As a consequence, the dimensionality of Gabor feature is $U \cdot V \cdot W_d \cdot H_d$.

In conclusion, rooted from Gabor filter, Gabor feature shows its advantages in the following aspects: (1) gathering information regarding both space and frequency; (2) generating a great variety of image representation; (3) orientation selective and according with head pose estimation purpose.

Histogram of Oriented Gradients

HOG has gained a wide application in computer vision field. Proposed by Navneet Dalal and Bill Triggs in [10], HOG was originally designed for human detection. Later, HOG demonstrates its effectiveness in many computer vision tasks, such as texture classification [46], digit classification [19] and face analysis related tasks [14, 52].

As a gradient based feature, HOG is achieved by calculating distribution of gradient orientations. Firstly, an image is divided into multiple cells. Denote I as the original image, and C as a cell. For a cell, two gradient images, C_x and C_y, respectively along horizontal and vertical directions are calculated. C_x and C_y are obtained from convolution, expressed by $C_x = C * D_x$, and $C_y = C * D_y$, where $D_x = [-1, 0, 1]$, and $D_y = [-1, 0, 1]^T$. Based on C_x and C_y, magnitudes of the gradients are defined by $|G| = \sqrt{C_x^2 + C_y^2}$, and orientations of gradients are calculated with $\theta = \arctan \frac{C_y}{C_x}$.

Distribution of orientations is generated by casting gradient value for each pixel to a histogram, weighted by the magnitude. After that, local normalization for the histogram is carried out, to alleviate illumination and contrast change in an image. Finally, HOG feature of image I is generated from the normalized histogram.

HOG emphasizes the gradient and orientation information, which is significant in describing shape and geometrical information. Prior work [18, 68] showed that HOG reflects pose variation and serves as an effective image descriptor in head pose estimation task.

2.4.3 Machine Learning Techniques

Random Forest

Random forest algorithm is a reliable tool in multi-class classification and regression tasks. Prior study uncovers its powerfulness with laboratory data [40, 35]. Besides, random forest algorithm shows its effectiveness in the industrial world. One representative example is the application of random forests to the development of Kinect by Microsoft company [59].

A random forest is a representative example of ensemble methods. Ensemble methods are built on a group of base estimators, and each estimator relies on a subset of the original training set. Finally, base estimators are integrated to be an ensemble. Since our proposed method is classification based, in the following part, we focus on the ensemble of classifiers.

In general, there are two families of ensemble methods: boosting methods and averaging methods [24]. In boosting methods, base classifiers are learned successively and iteratively. Each classifier relies on a weighted training set in order to focus on poorly classified instances in previous iteration. Therefore, each classifier is dependent on its precursors. Finally, a boosted classifier is generated from a sequence of weak classifiers. On the contrary, averaging methods build base classifiers independently, and the training set of each base classifier is generated from the original training set through bootstrap sampling (Appendix C). In classification tasks, outcomes of base classifiers are averaged to generate the final result.

Random forests are typical averaging methods. A random forest is an ensemble by integrating multiple decision trees, with randomness coming from the construction of each tree. In a tree scheme, each instance is considered as a set of attribute-value pairs. A decision tree is a directed graph, where a non-leaf node specifies a decision function on the attributes, its descending branch represents a possible outcome, and each leaf node represents a unique class. Sending an instance down to a decision tree, a final classification result is reached through a number of decision rules. A simple example of a decision tree classifier is illustrated in Figure 2.4.

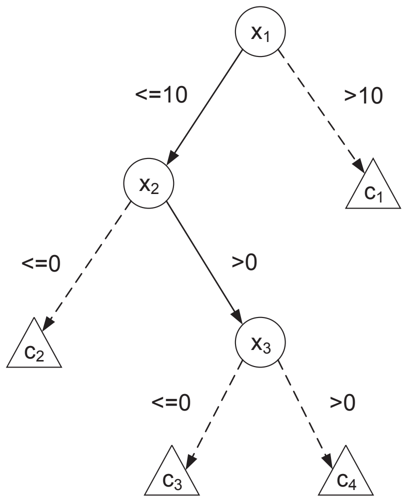


Figure 2.4: An example of a binary decision tree. Each instance has three attributes (x_1, x_2, x_3). Each non-leaf node represents a decision function, labeled by the attribute to be tested, and its descending edge denotes the test result. Four leaf nodes indicate four class labels.

A predominate method of decision tree learning is recursive partitioning. Sending all the training instances to the root node, a decision tree is grown from up to down. Partition of instances means splitting of nodes, and the splitting proceeds in a recursive way until reaching a stopping criterion. At last, the training instances are partitioned into multiple subsets, each corresponding to a leaf node and a distinct class label. To build a minimal tree that fits the training set, the attribute that best partitions the instances is selected at each splitting node. Well known metrics for splitting measurement include Gini impurity and information gain.

In a random forest, decision trees are learned in parallel, as shown in Figure 2.5. Each tree relies on a bootstrapped set from the original set. Besides, to choose the best attribute in node splitting, a subset of candidate attributes is randomly selected. Both bootstrap sampling and attribute selection introduce diversity among decision trees and randomness in a forest. In classification tasks, output of the forest is the majority vote of the outcomes generated by decision trees. Empirically, compared to a single decision tree, a random forest tends to achieve better performance and show less inclination of overfitting.

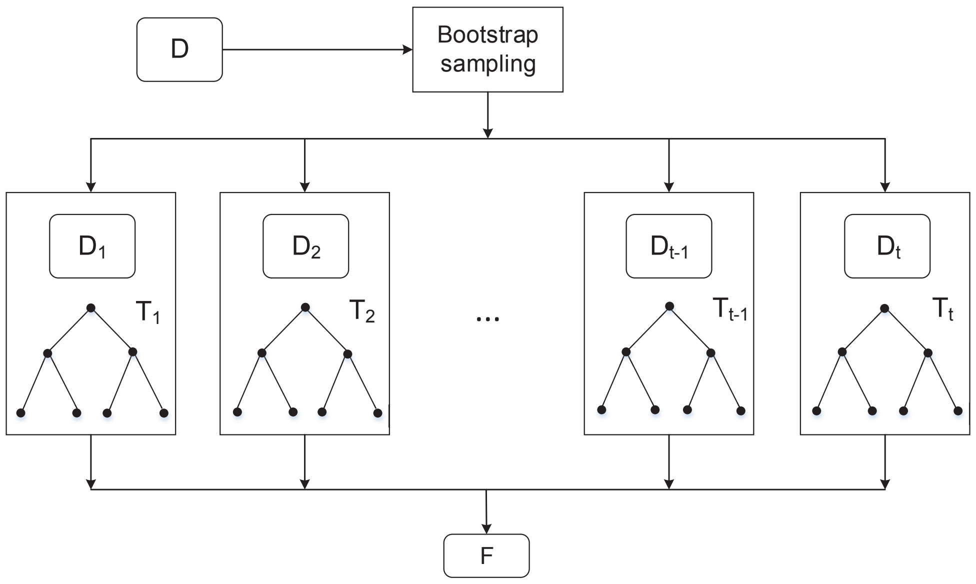


Figure 2.5: General structure of a random forest with totally t decision trees. D is the original training set; derived from D through bootstrap sampling, D_i is a training set for ith tree, where $i \in [1, t]$. By integrating all the individual trees, a forest F is generated.

Support Vector Machine

Support Vector Machine is a machine learning model which can be used in both classification and regression tasks [53]. For clarity, in this thesis, Support Vector Machine (SVM) denotes a tool specifically for building a classifier, while Support Vector Regression (SVR) is designed for regression.

A binary SVM classifier is described below. Suppose there is a set of training instances belonging to two classes, represented as $D = \{(x_1, y_1), ..., (x_N, y_N)\}$, where x_i represents ith instance, $y_i \in \{-1, 1\}$ indicating its label, and $i \in [1, N]$. SVM works by constructing a hyperplane to separate the two classes. Testing instances are then classified based on which side of the hyperplane they fall into.

Mathematically, a hyperplane can be expressed as $w^T x + b = 0$, where w is a normal vector to the hyperplane, and b is the bias. In many cases, training instances are not linearly separable. One solution is mapping the instances to a higher-dimensional space where they can be linearly separated. Denoting ϕ as the mapping function, the hyperplane is then represented by $w^T\phi(x) + b = 0$. To efficiently perform the classification, kernel function is then introduced, defined by $K(x_i, x_j) \equiv \phi(x_i)^T\phi(x_j)$.

There may be many hyperplanes that can separate instances from different classes, while the best one maximizes the margin, which denotes the distance between the hyperplane and the nearest instances of each class, as illustrated by Figure 2.6. SVM finds the best solution by solving a quadratic optimization problem, illustrated by

$$\min_{w,b,\varepsilon} \frac{1}{2}w^T w + C\sum_{i=1}^{N}\varepsilon_i \tag{2.6}$$

subject to

$$y_i(w^T\phi(x_i) + b) \geq 1 - \varepsilon_i \tag{2.7}$$

$$\varepsilon_i \geq 0 \tag{2.8}$$

where C is a regularization parameter, which trades off maximizing the margin and fitting the training instances; ε_i is a slack variable in soft margin, which controls the sensitivity to outliers.

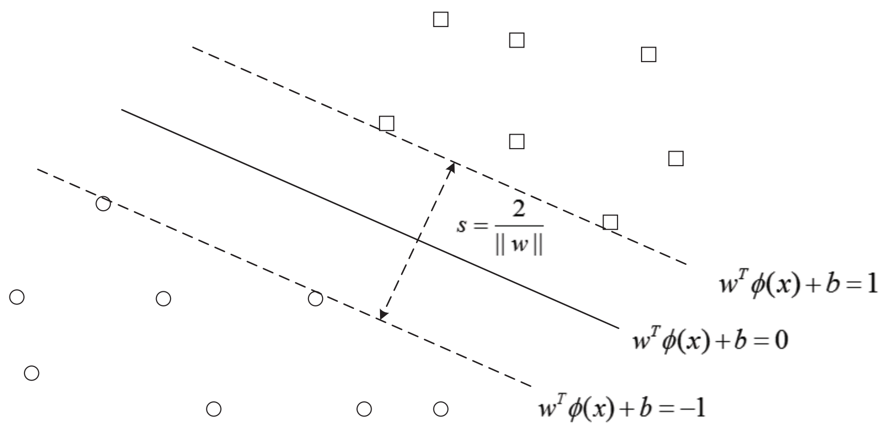


Figure 2.6: Maximum-margin hyperplane (the solid line) and the margin with distance s. Squares and circles represent instances of two classes.

In many classification tasks, there exist more than two distinct classes. A common solution is breaking down the problem into several binary classification tasks, in a one-vs-rest, or one-vs-one strategy.

In one-vs-rest, a classifier is built in terms of one class, with instances belonging to that class labeled as positive training data and all the rest instances as negative data. Given a testing instance, besides the estimated class label, each classifier also generates a confidence score representing the classification possibility. The label that corresponds to a highest score is the final classification result.

In one-vs-one, each classifier is built to distinguish two classes. Classification for an unlabeled instance relies on a voting scheme, i.e., the class with the highest number of positive predictions by binary classifiers is the final classification result.

Comparison of the two strategies has been conducted by Hsu and Lin [31]. They claim that two strategies achieve similar performance. In addition, training procedure of one-vs-rest is more complex and hence takes a huge time, while one-vs-one requires a much shorter training time and demonstrates more feasibility.

Chapter 3

Coarse-to-Fine Head Pose Estimation

In this thesis, we propose a coarse-to-fine method to estimate head pose described by yaw and tilt. By defining a frontal face to be with $0°$ yaw and $0°$ tilt, we are interested in yaw and tilt spanning within the range $(-90°, 90°)$, considering the fact that valid face part would be missing from the image under a pose out of the range. For an input face image, our method does not calculate the angles directly. Instead, we compare with the labelled images from a database to determine the angles. In addition, we discretize the pose space, therefore there are a finite number of head pose configurations, indicating head pose estimation is a classification task.

As an instance of supervised learning, classification requires training images to be annotated with labels. The label of a human face image is from the person's head pose. If yaw and tilt are estimated separately, two independent classifiers are trained, with yaw and tilt as the image label respectively. In the testing phase, two classifiers separately generate estimated yaw and tilt. In coarse-to-fine method, instead of estimating yaw and tilt in a separate way, we consider yaw and tilt as a whole and conduct combined estimation, given the fact that head pose is co-determined by both of them. Hence, for a face image, its label is represented by the vector $(yaw,tilt)$, which acts as the standard format for an image label in the following part of this thesis.

The following part of this chapter presents coarse-to-fine method in detail. Chapter 3.1 describes the coarse-to-fine estimation mechanism. Chapter 3.2 presents ROI detection module. Finally, coarse-to-fine method is summarized in chapter 3.3.

3.1 Coarse-to-Fine Estimation

Rather than directly performing classification in the pose space, we propose a two-step estimation mechanism, i.e., coarse-to-fine estimation. An underlying assumption of coarse-to-fine estimation is the clustering in the original pose space, following the principle that neighboring angles are clustered together and form a subspace.

Figure 3.1 illustrates the clustering results. Original yaw space is divided into several subspaces, expressed as

$$Y = \{y_1, y_2, \ldots, y_M\} \tag{3.1}$$

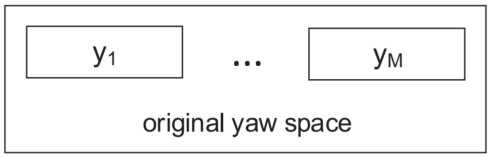
Similarly, division of tilt space is given by

$$T = \{t_1, t_2, \ldots, t_N\} \tag{3.2}$$

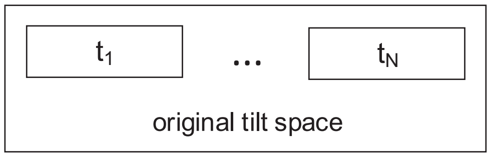
where Y and T respectively denote the original yaw space and tilt space; y_i $(i \in [1, M])$ represents the yaw subspace, and t_j $(j \in [1, N])$ represents the tilt subspace; M and N represent the number of subspaces for yaw and tilt. Since we employ combined estimation of yaw and tilt, pose configurations across the subspaces come from Cartesian product of Y and T, described by

$$Y \otimes T = \{(y_i, t_j) \mid y_i \in Y \text{ and } t_j \in T, \forall i = 1, ..., M,\ j = 1, ..., N\} \tag{3.3}$$

where $\otimes$ denotes Cartesian product operation.



(a)



(b)

Figure 3.1: Pose space division from clustering

With deep insight into each yaw subspace, y_i is a set consisting of m_i elements from the original yaw space:

$$y_i = \{y_{i1}, y_{i2}, \ldots, y_{im_i}\} \tag{3.4}$$

Similarly, each tilt subspace t_j has n_j unique elements:

$$t_j = \{t_{j1}, t_{j2}, \ldots, t_{jn_j}\} \tag{3.5}$$

Cartesian product of y_i and t_j is denoted as

$$y_i \otimes t_j = \{(y_{ip}, t_{jq}) \mid y_{ip} \in y_i \text{ and } t_{jq} \in t_j, \forall p = 1, ..., m_i,\ q = 1, ..., n_j\} \tag{3.6}$$

From the clustering process, the original pose space can be decomposed in a hierarchical structure, as illustrated in Figure 3.2. Starting with a node which represents a set of all the pose configurations in the original space, the structure consists of three layers. In particular, the clustering process gives rise to the second layer, where each node can be viewed as a subset in the pose space. Eventually, in the third layer, all the pose configurations are separated.

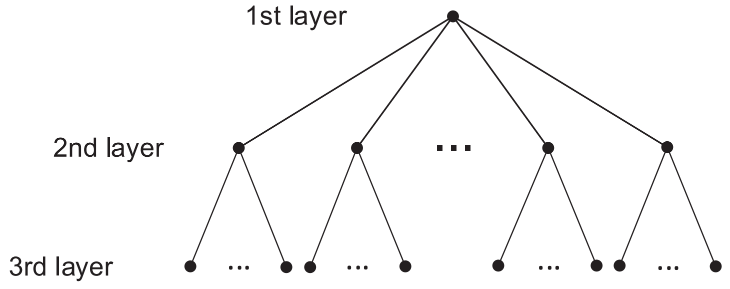


Figure 3.2: Hierarchical decomposition of the pose space. The first layer represents a set of original pose configurations; each node in the second layer represents (y_i, t_j); each node in the third layer represents (y_{ip}, t_{jq}), with $i \in [1, M]$, $j \in [1, N]$, $p \in [1, m_i]$, $q \in [1, n_j]$.

Based on the hierarchical representation of pose configurations, a two-step estimation mechanism—coarse-to-fine estimation is proposed. Specifically, coarse estimation identifies a unique path from the first layer to the second layer, and fine estimation further extends the path to the last layer.

In the training phase, training images are clustered into several isolated groups, based on yaw and tilt angles given as the ground truth. Each group corresponds to a unique (y_i, t_j). Images from a group share the same group Id and generate a unique class. Then a classifier is trained to distinguish these classes. After that, images within the same group are annotated with distinct within-group Ids and form multiple subclasses. Images in each subclass correspond to a unique (y_{ip}, t_{jq}), and therefore cannot be further partitioned. To discriminate these subclasses, a classifier is trained using images within the group as training instances. It is apparent that in each group, a classifier is needed. Ultimately, a unique vector (*group Id, within-group Id*) denotes a unique pose configuration in the original pose space.

In the testing phase, for an input image, its group Id is firstly estimated, identifying a specific node in the second layer based on Figure 3.2. In other words, the first estimation step outputs a coarse estimation result and specifies a set of candidate pose configurations. Therefore, the classifier in the first step is called coarse classifier. Afterwards, within-group

Id of the input image is estimated. Compared to the first estimation step, the second step generates a refined result. Correspondingly, the classifier in the second step is called fine classifier. Overall, the two-step estimation mechanism is denoted as coarse-to-fine estimation. Finally, from two estimation steps, a unique vector (*group Id, within-group Id*) is generated, from which the label of the input image can be inferred.

In this thesis, training and testing phases in terms of the coarse classifier are denoted as the coarse stage, while training and testing in terms of a fine classifier are denoted as the fine stage. A general framework of coarse-to-fine estimation is illustrated in Figure 3.3.

The reason for employing a two-step estimation mechanism is that there are usually too many labels in a head pose estimation task. According to Equations 3.1 to 3.6, there are totally $\sum_{i=1}^{M} m_i$ yaw angles, and $\sum_{j=1}^{N} n_j$ tilt angles. Since we employ combined estimation, the number of unique labels is $(\sum_{i=1}^{M} m_i) \cdot (\sum_{j=1}^{N} n_j)$. Direct estimation in the original pose space requires training a classifier to discriminate these classes. On the contrary, in the coarse-to-fine estimation mechanism, distinct classes are much fewer. In the coarse stage, the number of distinct classes is $M \cdot N$; in the fine stage, the number of classes is $m_i \cdot n_j$. Hence, coarse-to-fine estimation can greatly reduce the complexity of each classifier and improve its performance. Furthermore, compared to independent estimation of yaw and tilt, coarse-to-fine method can prevent some unlikely combinations of the two, thereby enhancing the overall performance.

Next, we conduct a detailed analysis of coarse classifier (Chapter 3.1.1) and fine classifiers (Chapter 3.1.2). Each part is presented from two aspects—image descriptor and machine learning technique.

3.1.1 Coarse Classifier

Gabor Feature

In both training and testing phases, generating image descriptors is always the first step. In terms of coarse classifier, Gabor feature is employed to represent the images. As stated in Chapter 2.4.2, Gabor feature is sensitive to orientation, due to a Gabor filter bank and a Gabor image array. Figure 3.4 illustrates real parts and magnitudes of a Gabor filter bank of size 5×8. As shown in Figure 3.4a, Gabor filters display various orientations. Besides, filters in each column show an increasing scale.

Generated by a Gabor filter bank shown in Figure 3.4 and a face image in Figure 3.5, a Gabor image array is illustrated in Figure 3.6, which indicates that each filter has its own focus in representing an image. Generally speaking, a filter with a large scale highlights the overall contour and suppresses personal characteristics. On the contrary, a filter with a small scale puts more weight on local details and responds well to difference in orientation, making facial components stand out.

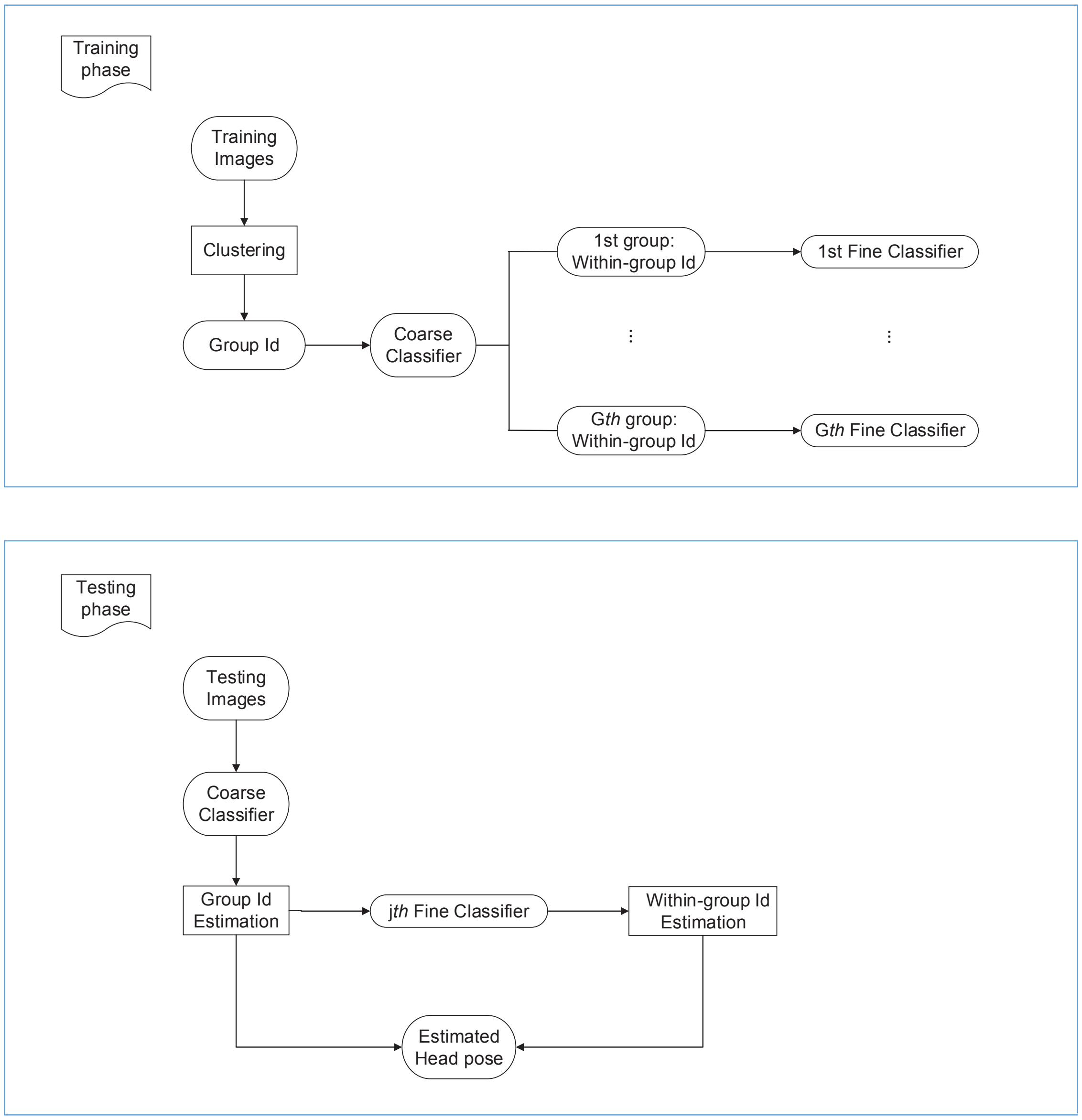
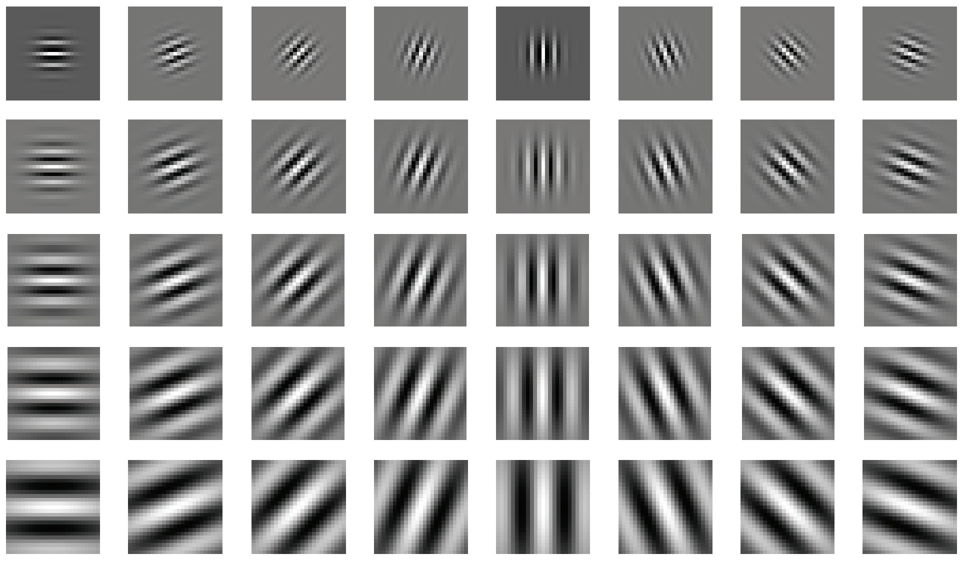
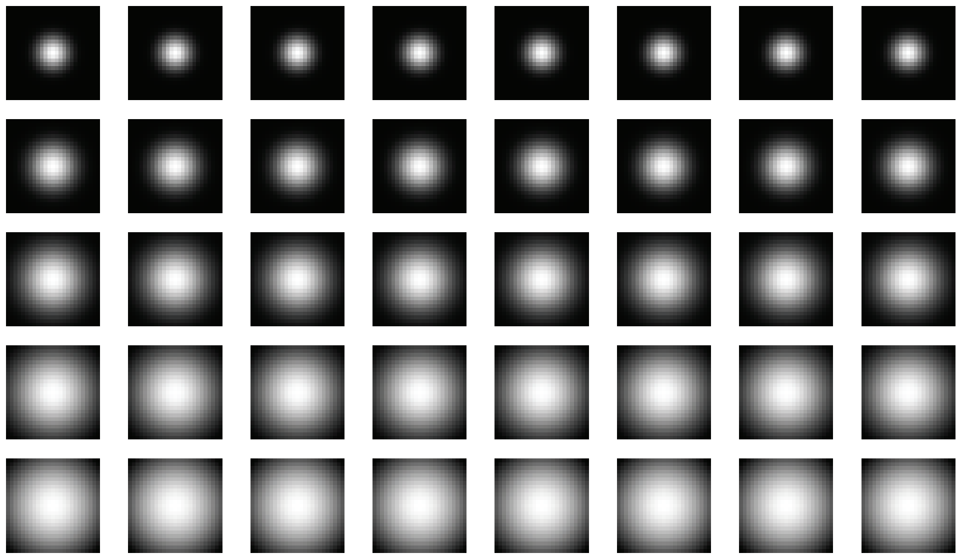


Figure 3.3: General framework of coarse-to-fine estimation. In the training phase, G denotes the number of distinct groups; based on Equations 3.1 to 3.3, G is equal to $M \cdot N$. In the testing phase, from the estimated group Id, jth fine classifier is selected. Each rounded box represents data and each rectangular box represents process.



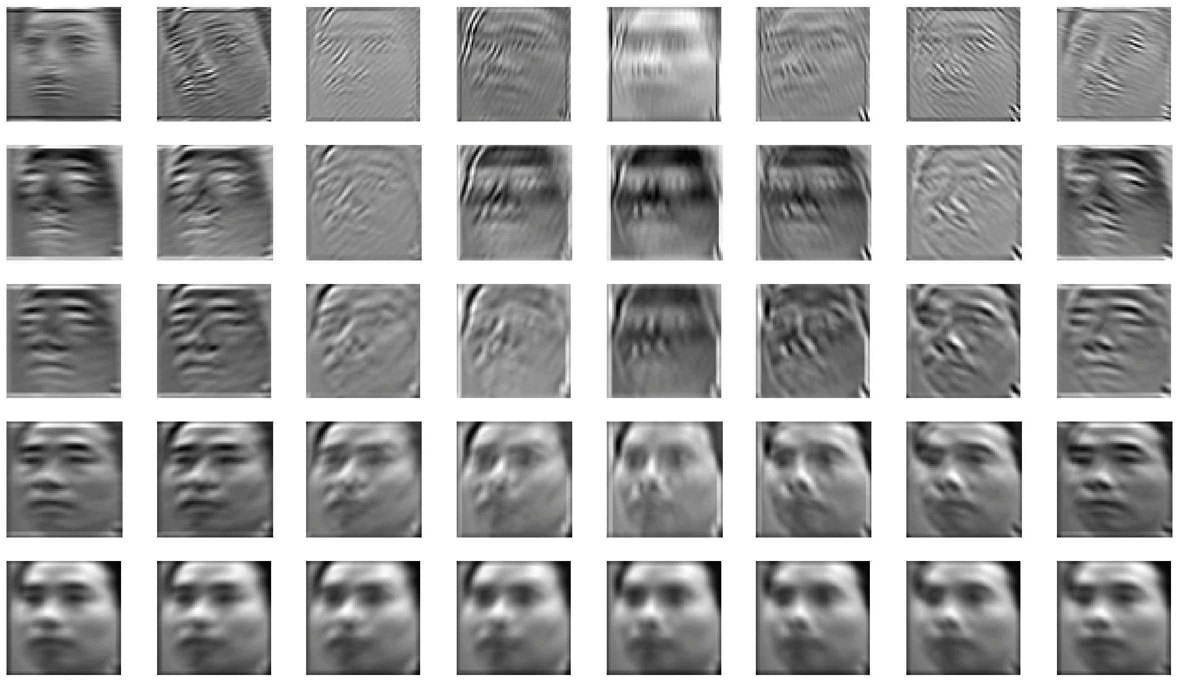
(a)



(b)

Figure 3.4: (a) Real parts of Gabor filters; (b) Magnitudes of Gabor filters. Based on Equations 2.1 to 2.4, parameters for the filter array are set as follows: $f_{\max} = 0.25$, $\gamma = \eta = \sqrt{2}$, $m = n = 25$, and, $U = 5$, $V = 8$.

Figure 3.5: An example face image


(a)

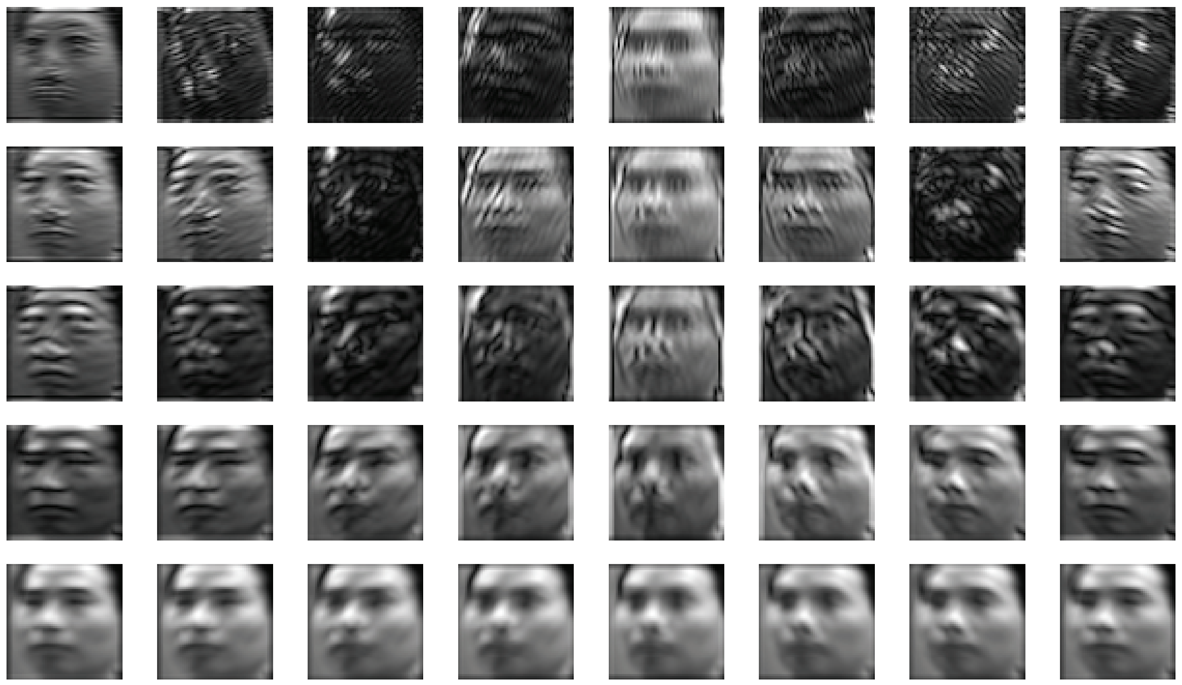

(b)

Figure 3.6: (a) Real parts of Gabor images; (b) Magnitudes of Gabor images.

Random Forest

With Gabor feature as the image descriptor and group Id as the image label, a coarse classifier is trained to discriminate groups. Random forest algorithm is employed in the training phase, considering its capability to handle large amounts of data and effectiveness demonstrated in many other work.

In growing each decision tree, information gain is the metric for selecting the best attribute in node splitting, expressed as

$$I_n^a = Entropy(parent) - AverageEntropy(children) \tag{3.7}$$

$$= H(S_n) - \sum_{k \in [L,R]} (|S_n^k|/|S_n|)H(S_n^k) \tag{3.8}$$

where n is the node to be split, a is an attribute under measurement, and k represents child of n. H represents entropy. For a set S composed of multiple instances, $H(S)$ is calculated by $\sum -p\log_2 p$, with p representing the probability of a class. S_n is a set at n, while S_n^L and S_n^R respectively represent the left and right subsets; $|\cdot|$denotes the number of instances in a set.

At a node to be split, information gain measures how well an attribute would partition the instances by an expected entropy reduction. High information gain means less impurity in partitioned instances. Thus the attribute which maximizes information gain is selected as the best one.

3.1.2 Fine Classifier

Histogram of Oriented Gradients

On top of the coarse stage in which classification of different groups is achieved, training a fine classifier is aimed to further classify instances within a group. In each group, distinct labels cover a small range of the original pose space, implying high similarity among classes in the fine stage. As a result, extracting image descriptors to emphasize the pose variation among different classes is significant.

From a detailed description in Chapter 2.4.2, HOG is computed by extracting orientation information, which is critical in describing the head pose. Orientation information incorporated in HOG is illustrated in Figure 3.7. In addition, HOG shows invariance to illumination change. Therefore, HOG is used to represent images in the fine stage.

Support Vector Machine

The machine learning technique in training a fine classifier is a multi-class SVM with one-versus-one strategy, given the fact that there are usually more than two distinct classes within a group.

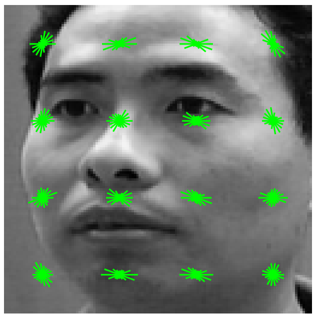


Figure 3.7: Orientation information in HOG. Within the face image from Figure 3.5, green lines indicate orientation generated from gradients. For each cell, there are totally 9 bins for orientation distribution.

3.2 ROI Detection

Points in a detected face take on various degrees of importance in terms of estimating head pose, based on the following considerations:

- irrelevant and distractive information introduced in the face detection part (discussed in Chapter 1)
- tight correlation between head pose and geometrical characteristics of the face image (discussed in Chapter 2.4.2)

Therefore, in head pose estimation task, we want to eliminate undesired information and focus on the area containing the most important image points rather than the whole image. To this end, we propose a ROI detection module, and head pose is estimated from ROI within a face image through coarse-to-fine estimation (Figure 3.8).

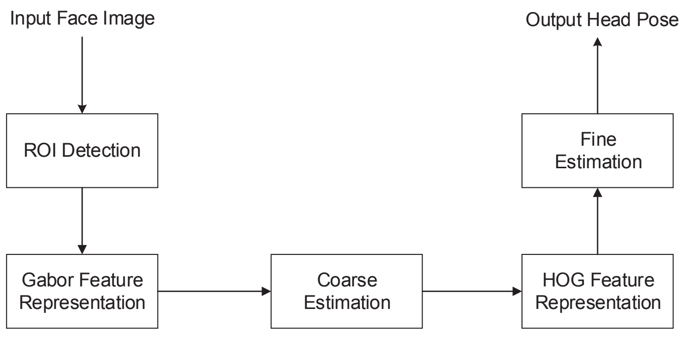


Figure 3.8: Combination of ROI detection and coarse-to-fine estimation

In the ROI detection module, the most important image points, i.e., keypoints, are firstly selected from the face image. ROI is then generated from locations of the keypoints. Keypoint selection is achieved with the help of a sequence of random forests. In building a forest, each variable corresponds to a distinct dimension in the feature space. In addition to classifying instances, a random forest provides a measurement of variable importance. Based on variable importance score, variable selection is conducted in an iterative strategy. After that, indexes of the selected variables are used to locate keypoints in an image.

3.2.1 Variable Importance Measurement

In a random forest, the most commonly used metric for evaluating variable importance is called permutation importance [27], generated by out-of-bag data.

Chapter 2.4.3 discusses that the training set of a decision tree in the forest is a bootstrapped set of the original training set. Out-of-bag data for each decision tree consists of instances in the original set but absent from the bootstrapped set. Importance score for a given variable j is then calculated from accuracy decrease generated by out-of-bag data. In detail, for each tree, its out-of-bag instances are firstly sent down, generating a classification accuracy. After that, the values of variable j in out-of-bag instances are randomly permuted, and out-of-bag instances after permutation are then classified by the decision tree, generating a new accuracy and a decrease compared to the original one. Finally, the mean accuracy decrease over all the decision trees in a forest is the importance score for variable j.

Repeating the above process for each variable, importance scores for all the variables are generated. Generally speaking, a variable with a higher score is considered to be more important. It is the fundamental assumption in variable selection.

3.2.2 Variable Selection

Given image descriptors of dimension L, the most important L^* variables are to be selected. Obviously, $0 < L^* < L$.

Inspired by earlier work conducted by Díaz-Uriarte et al. [15], variable selection involves variable importance score ranking and an iterative variable introduction. Initially, through the random forest from the coarse stage, importance scores for totally L variables are computed and ranked from high to low. Variable introduction proceeds as follows. Given a defined percentage p $(0 < p < 1)$, variables with the first $\lfloor p * L \rfloor$ importance scores are selected, while the remaining ones are discarded. If the length of selected variables is greater than L^*, a new random forest based on image descriptors with selected variables is built, followed by a new iteration of variable introduction. The procedure repeats until the length of selected variables is equal to L^*. Algorithm 1 summarizes the variable selection process.

Instead of directly selecting the first L^* variables in the initial step, iterative variable introduction selects the important variables in a gradual fashion. As a result, selected variables are sifted by a sequence of random forests and are thereby more convincing.

Algorithm 1 Variable Selection

```
while the length of selected variables is greater than $L^*$ do
    build a random forest based on descriptors with current variables;
    compute variable importance scores;
    rank the scores from high to low;
    keep variables with the first $p$ percentage scores and discard the rest;
    record indexes of the selected variables in the original image descriptor.
end while
return A sequence of selected variables with length $L^*$
```

3.2.3 Keypoint Selection

In variable selection, indexes of the selected variables are recorded simultaneously, acting as indicators of keypoint locations within the face image.

The process of locating keypoints is shown in Figure 3.9. In the coarse stage, the image descriptor for representing a face image is generated by concatenating pixels from magnitudes of Gabor images. As a result, making use of indexes of the selected variables in the image descriptor, critical points within the Gabor images are firstly traced. Then the locations of critical points within the Gabor images are used to get keypoint locations in the original image. Since a face image generates a Gabor image array, critical points within different Gabor images may overlap, hence generating keypoints in the original face image with multiple weights.

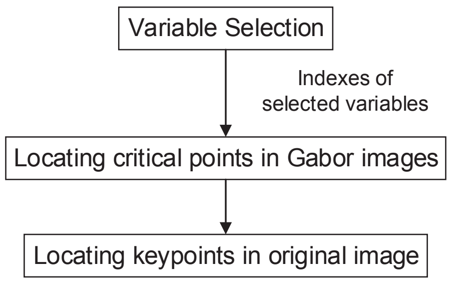

Variable Selection
Indexes of selected variables
Locating critical points in Gabor images
Locating keypoints in original image

Figure 3.9: Procedure of locating keypoints in the original image

3.2.4 ROI Detection

Given the selected keypoints in a face image, a minimum rectangle incorporating all the keypoints is generated. Meanwhile, the other one is formed by keypoints with multiple weights. An example is illustrated in Figure 3.10.

From Figure 3.10, we can make the following conclusions: (1) keypoints are mostly distributed in the face part; (2) quite a number of keypoints are assigned with multiple weights; (3) the blue rectangle excludes the background and hair, which are irrelevant to a

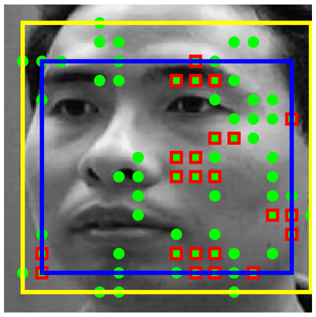


Figure 3.10: Keypoints in an example face image. Solid green circles represent selected keypoints, and red squares represent keypoints with multiple weights; the yellow rectangle and the blue one are generated by green circles and red squares respectively.

person's head pose. To eliminate undesired information within a face image, we define ROI as the region generated by keypoints with multiple weights.

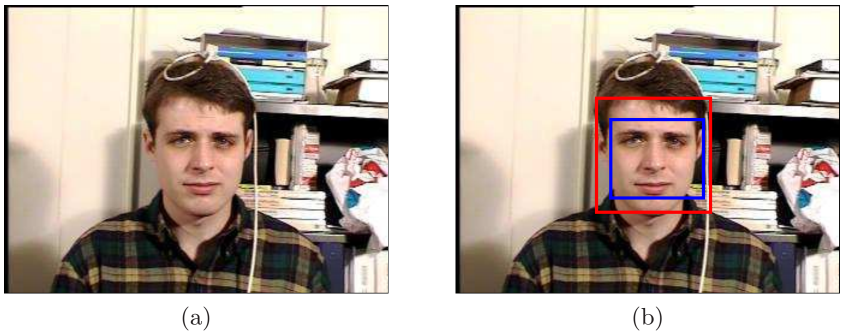


Figure 3.11: (a) a sample frame from Boston University dataset [37], where a person is wearing a 3D magnetic tracker to collect ground truth for head pose; (b) the red rectangle indicating the detected face by Viola-Jones algorithm [65], and the blue rectangle indicating the detected ROI.

3.3 Summary

In summary, we propose a coarse-to-fine head pose estimation method, by breaking down the perplexing estimation task into two progressive steps. Moreover, ROI detection achieves the goal of eliminating irrelevant information and focusing on the significant part within a face image.

Chapter 4

TV Viewers' Behavior Analysis

In this chapter, we apply the proposed coarse-to-fine head pose estimation method to analyze TV viewers' behaviors, which are under surveillance by a RGB camera mounted on top of the TV screen. This kind of analyses is useful for some applications, such as marketing research. For example, it can be used to study the effectiveness of ads in TV broadcasting. We design our application to be robust, reflected in the following aspects:

- applicable for both single-person and multi-person scenarios;
- no restriction for people's movement.

Under the fundamental assumption that head pose can be used to infer the visual attention, a TV viewer's head pose change over time reflects his inclination to the TV programme. Thus our application relies on extracting TV viewers' head pose information from video recordings. Processing video data needs integrate multiple tasks. Figure 4.1 illustrates the flowchart of analyzing TV viewers' behaviors.

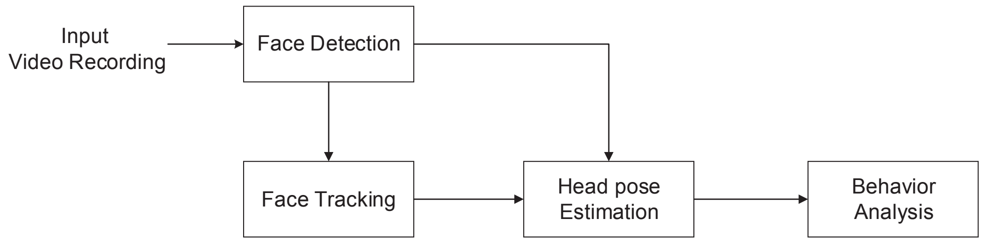


Figure 4.1: Flowchart of analyzing TV viewers' behaviors

Four components are contained in the application: (1) initially detecting face locations and assigning face Ids; (2) tracking faces in subsequent frames; (3) estimating head poses from face locations in each frame; (4) analyzing TV viewers' behaviors. Among the four components, face detection is introduced in Chapter 2.4.1, and head pose estimation relies on coarse-to-fine estimation method presented in Chapter 3. Thus in the following part, we focus on face tracking and behavior analysis.

4.1 Face Tracking

Given the face detection result, the goal of face tracking is to find face trajectories through a number of frames. Stably and accurately tracked faces are crucial to the following head pose estimation step. Besides, face tracking is expected to be efficient in order to build a real-time application. From our knowledge, the KLT algorithm is well known for its competitive performance and efficiency. Therefore, KLT is employed in our application to conduct face tracking.

The first step in the KLT algorithm is extracting feature points from the initial position. Under the assumption that brightness constancy and small appearance change exist between two adjacent frames of a video [67], KLT finds the feature point correspondence across frames. In our application, in order to emphasize stable feature points in the face area, face tracking is achieved by a weighting function of feature points. Furthermore, to eliminate invalid tracking result caused by occlusion, we propose a consistency checking module following each tracking process.

4.1.1 Feature Point Detection

In order to be tracked reliably, feature points should have richly structured neighborhood. Proposed by Tomasi and Kanade in [63], feature points are image corners which have strong gradients in at least two directions. To find image corners, a structure matrix G is defined by

$$G = \iint_W \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} dxdy \tag{4.1}$$

where I_x and I_y are gradients of image I, given by $I_x = \partial I(x,y)/\partial x$, and $I_y = \partial I(x,y)/\partial y$; W is the size of an image window.

Eigen decomposition of G is used to analyze the local pattern within the window. Generally speaking, two small eigenvalues of G indicate a homogenous pattern; a big eigenvalue and a small one imply an unidirectional pattern, i.e., an edge; two large eigenvalues indicate a corner. By defining the threshold of the minimum eigenvalue, corner points can be detected in their local windows.

4.1.2 Feature Point Tracking

Consider a corner point (x, y) in image I. The goal of tracking is to find the location $(x + v_x, y + v_y)$ in image J such that $I(x, y)$ and $J(x + v_x, y + v_y)$ are matched. The vector (v_x, v_y) is called the velocity at (x, y), denoted by $\mathbf{v}$. Similar to feature detection, KLT defines the matching in a local window. An error function ε in terms of $\mathbf{v}$ is expressed by

$$\varepsilon(\mathbf{v}) = \varepsilon(v_x, v_y) = \iint_W [I(x, y) - J(x + v_x, y + v_y)] dxdy \tag{4.2}$$

By taking Taylor expansion of $J(x + v_x, y + v_y)$ and then differentiating ε with respect to $\mathbf{v}$, the optimal $\mathbf{v}$ is obtained.

$$\mathbf{v}_{opt} = G^{-1}(\iint_{W} ((I(x,y) - J(x,y))[I_x, I_y]^T)dxdy) \tag{4.3}$$

Equation 4.3 can generate reliable tracking result when there is small pixel motion between I and J. To allow large motion between them, pyramid representation of images is employed, and feature point tracking is carried out from the lowest resolution level up to the original one (Figure 4.2). An additional step in KLT tracking is verifying the tracked points [58]. By checking the local window similarity, incorrectly tracked points are dropped.

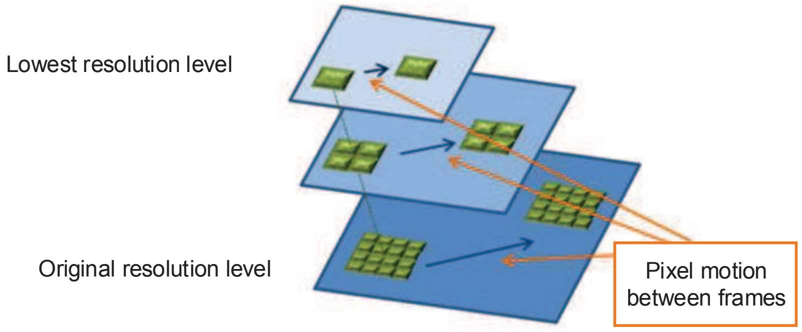


Figure 4.2: Image Pyramid in KLT tracking

4.1.3 Face Tracking

In face tracking, feature points are detected from the initial face location, and outputs of KLT are velocities of tracked feature points. To describe the motion of a face between two adjacent frames, an intuitive idea is averaging all the velocities in order to reflect a general moving trend. In our application, instead of treating feature points equally, face motion is calculated by a weighted average of point velocities. Mathematically, the motion vector between two adjacent frames is defined by

$$MV(f_i, f_{i+1}) = \sum_{j=1}^{n} w_1\mathbf{v}_1 + w_2\mathbf{v}_2 + ... + w_n\mathbf{v}_n \tag{4.4}$$

where f_i and f_{i+1} represent the previous frame and the current frame respectively; n denotes the number of tracked feature points; for jth ($j \in [1, n]$) point, $\mathbf{v}_j$ represents the estimated velocity , and w_j represents the weight; MV is the motion vector from f_i to f_{i+1}. The weight assigned to a feature point is given by

$$g_j = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(x_j - x_c)^2 + (y_j - y_c)^2}{2\sigma^2}) \tag{4.5}$$

$$w_j = g_j \Big/ \sum_{j=1}^{n} g_j \tag{4.6}$$

where (x_j, y_j) is the coordinate of the feature point, and (x_c, y_c) is the coordinate of the centroid of all the feature points, given by $x_c = \frac{1}{n}\sum_{i=1}^{n} x_j$, $y_c = \frac{1}{n}\sum_{i=1}^{n} y_j$; σ is the standard deviation; g_j represents an initial Gaussian weight; after normalization, w_j represents the weight assigned to jth feature points.

Based on Equations 4.5 and 4.6, the weight of a feature point is associated with the point location. A feature point closer to the centroid is considered to be more stable in representing a face, thus assigned with more weight. On the contrary, the peripheral point is considered to be unstable in face tracking, therefore with less weight. Figure 4.3 illustrates the weights of feature points in a face.

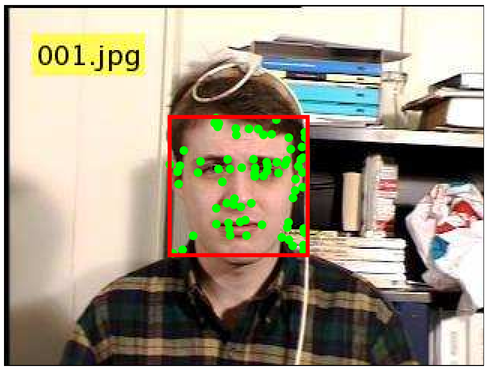
Compared to simply averaging the feature point velocities, the weighting function is able to improve consistency and accuracy in face tracking. Finally, from the face location in f_i and the estimated motion vector, the face location in f_{i+1} can be identified. Examples of tracking results can be found in Figure 4.4.

Equation 4.4 defines the trajectory for a single face. In multi-person scenarios, faces of different people are tracked in parallel. Occlusion between people may occur in multi-face tracking, making part of or the whole face invisible. As a result, the tracked face might be an invalid input in terms of head pose estimation. Some examples of invalid tracking results caused by occlusion are shown in Figure 4.5.

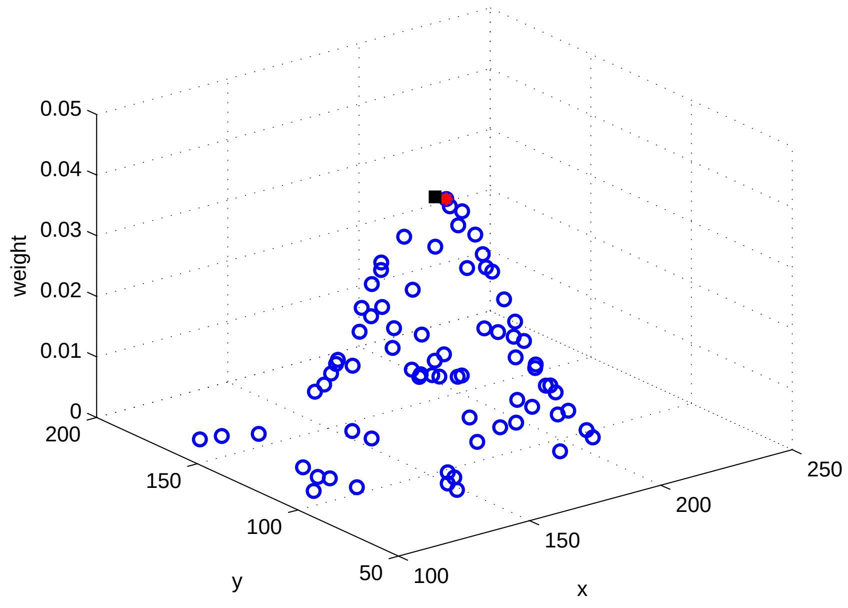
When occlusion happens, the consistency between two adjacent frames is broken. Only a small number of feature points would be tracked and their spatial layout might be disrupted. In addition, the tracked face contains only part of the original face area. To detect and then eliminate invalid tracking results, we propose to check consistency after conducting face tracking from f_i to f_{i+1}. Three criteria for identifying a valid tracking result are defined from different points of view, listed as follows:

1. the ratio between the number of feature points in f_{i+1} and f_i is above a threshold;
2. the difference between the sum of pairwise distances of tracked points in f_{i+1} and f_i is within a range;
3. the difference between joint color and texture histogram of faces in f_{i+1} and f_i is within a range.

The first two criteria are based on feature points, and the third one is aimed to check the global feature of the tracked face, represented by color and texture histogram. On one hand, color histogram describes color distribution of face pixels. On the other hand, texture histogram reveals the intensity variation and incorporates the spatial information which is

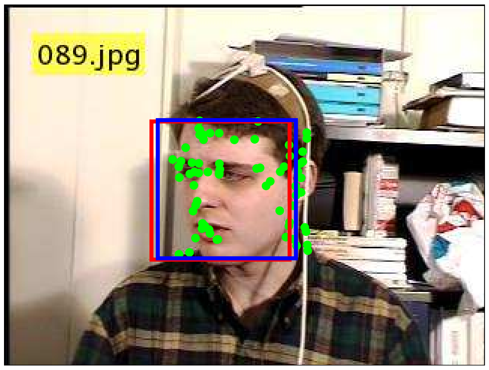


(a)

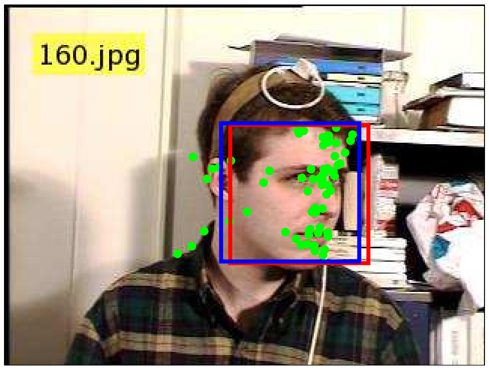


(b)

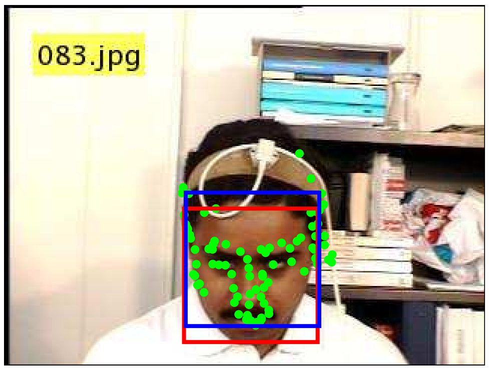
Figure 4.3: (a) Face and feature points within a sample frame from Boston University dataset [37]. The red rectangle indicates the detected face, and solid green circles represent its feature points by KLT; (b) Weights of feature points. Coordinates of feature points are defined in the x-y plane, and the feature weights are shown on the z axis. The red star marks the feature point with the maximum weight, and the black rectangle corresponds to the centroid of all the feature points.



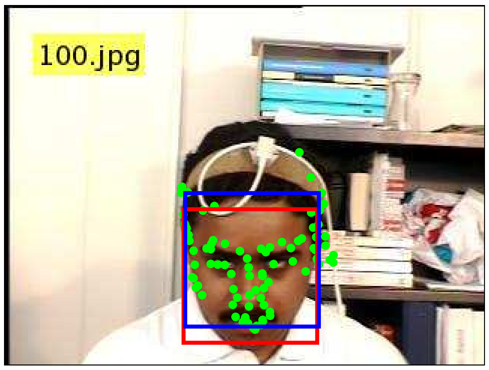
(a)



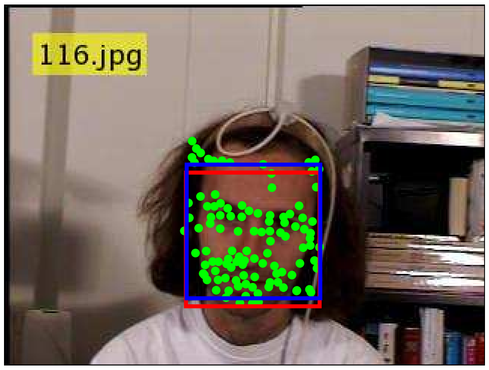
(b)



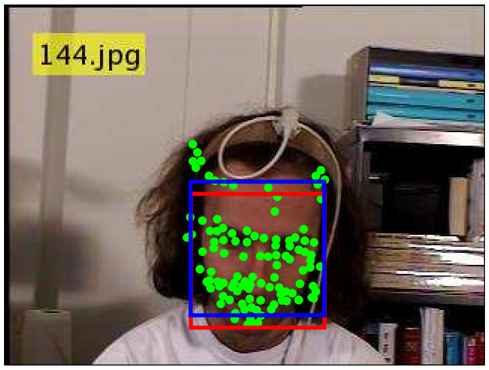
(c)



(d)

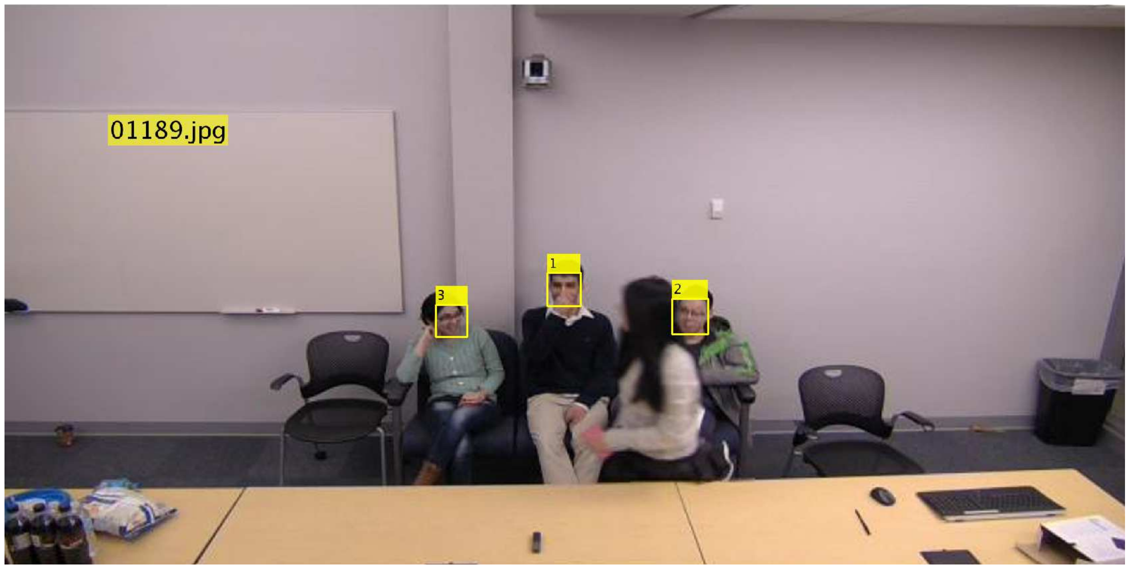


(e)

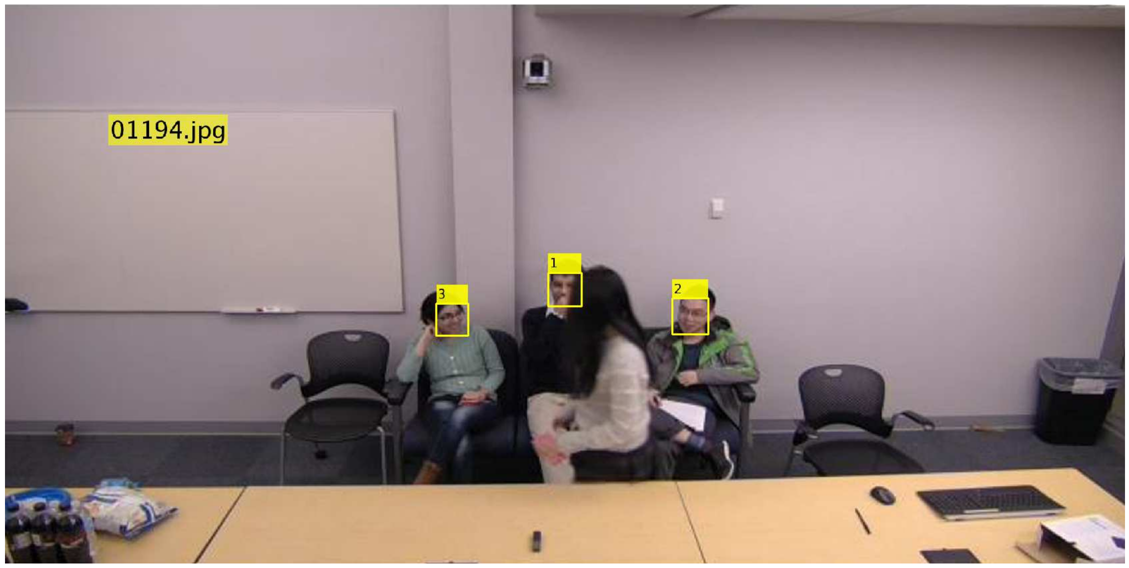


(f)

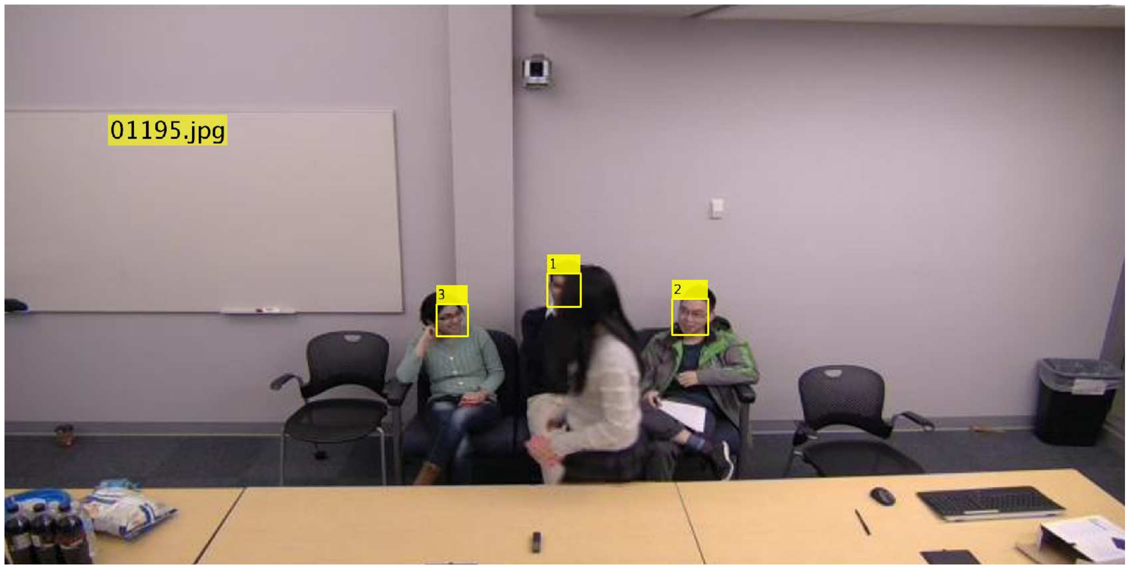
Figure 4.4: Comparison of tracking results with and without weighting function. Solid green circles represent tracked feature points; red rectangles indicate tracked faces using the weighting function; blue rectangles indicate tracked faces by treating feature points equally. Red rectangles capture the face motion more accurately, while blue rectangles are more affected by peripheral feature points.



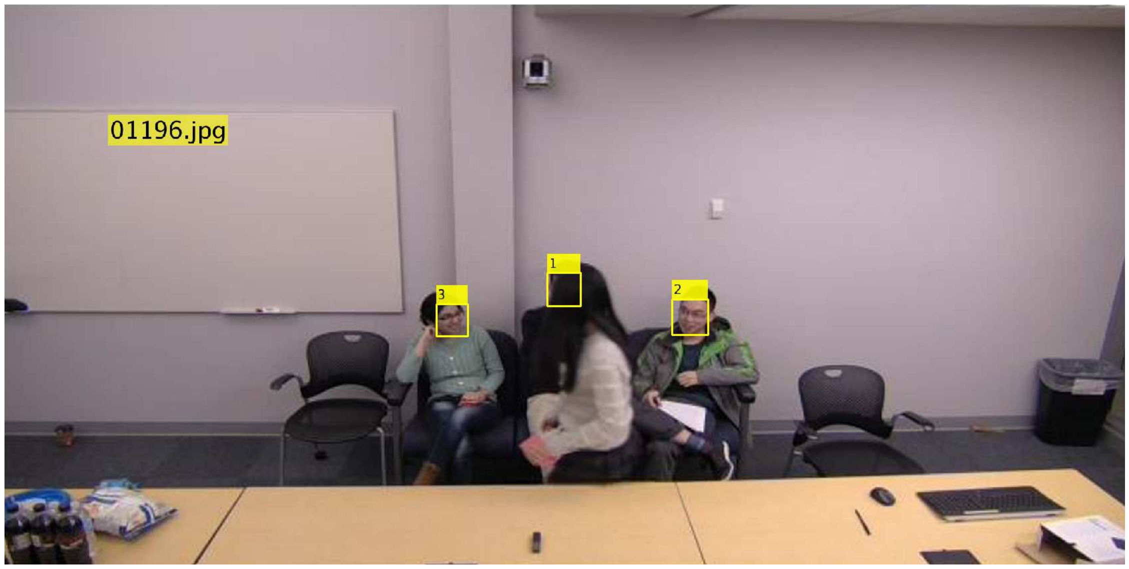
(a)



(b)



(c)



(d)

Figure 4.5: Illustration of invalid tracking results by occlusion. In (a), three faces are detected and assigned with unique face Ids; In (b), (c) and (d), the face with 1*st* Id is occluded, thus generating invalid detection results.

absent in color histogram. In our application, color information is from RGB space, and texture information is captured by LBP (Appendix D.1).

By checking the three criteria (Appendix D.2), we determine if the tracking from f_i to f_{i+1} is valid or not. The valid tracking result is kept and subsequent tracking goes on, while the invalid tracking result is deleted. After deleting intermediate tracking result, face re-detection is needed in order to detect the subsequent face locations. In summary, the flowchart of face tracking across frames is illustrated in Figure 4.6.

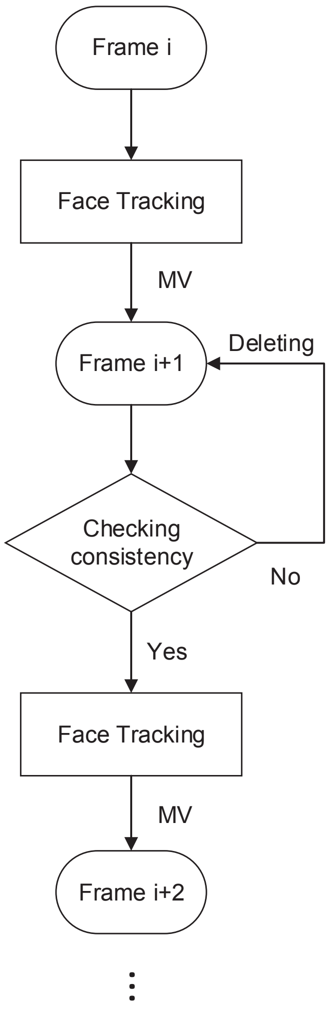


Figure 4.6: Flowchart of face tracking across frames

4.2 Behavior Analysis

Given face locations in each frame, our goal is to analyze TV viewers' behaviors. In this thesis, TV viewers' behaviors are divided into two classes, described by focused and unfocused respectively. With a camera mounted on top of the TV screen, frontal faces are captured when viewers are looking at the TV screen. On the contrary, non-frontal faces indicate viewers are distracted and unfocused.

Head pose information extracted from faces is a reliable resource in behavior analysis, and significant head pose change implies a nonnegligible head movement. We initially select a base frame in which the viewers are assumed to be looking at the TV screen. In the following frame sequences, head pose is estimated, and relative head pose change compared to the base frame is calculated. By setting a change threshold, unfocused viewers can be identified.

For each person in the scene, the video recording reflects the continuous pattern of head movement. However, coarse-to-fine estimation method works in a discretized pose space, thus incapable of detecting delicate head pose change. In addition, coarse-to-fine estimation method applies to face images under head poses within a range. Nevertheless, there is no restriction for a person's head movement. In order to identify a person's behavior even when his face is gradually out of view, we make use of the motion vector of the face to capture the head movement.

Based on Equation 4.4, motion vector from the base frame to the current one is calculated by

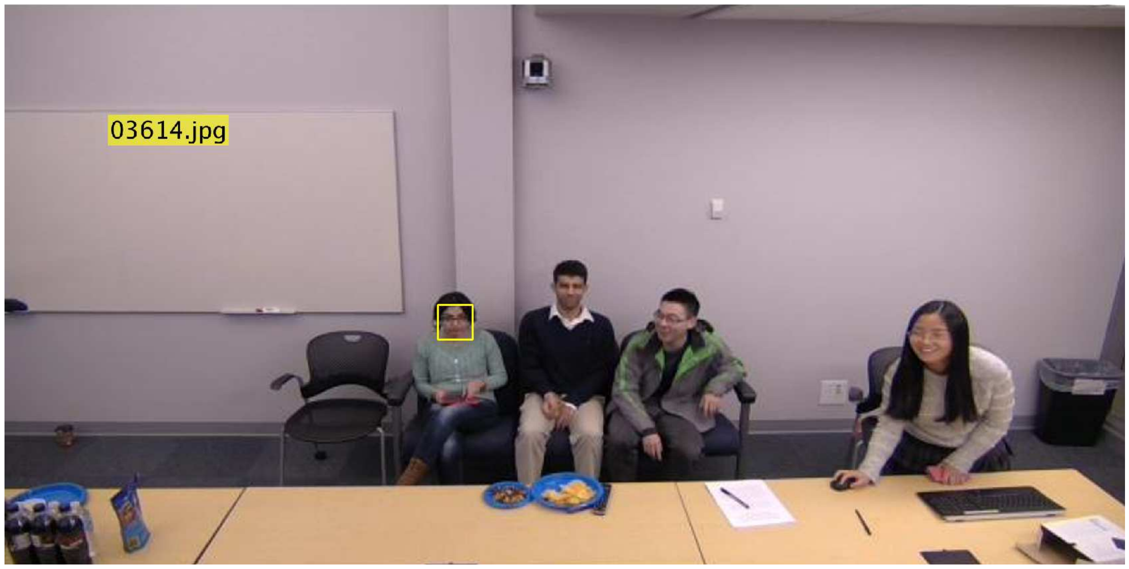
$$MV(f_s, f_c) = \sum_{i=s}^{c-1} MV(f_i, f_{i+1}) \tag{4.7}$$

where f_s and f_c denote the base frame and current frame respectively ($c > s$); MV denotes the motion vector; $MV(f_i, f_{i+1})$ ($s \leq i \leq c-1$) is generated during face tracking process. From Equation 4.7, $MV(f_s, f_c)$ can be viewed as an accumulated motion vector between adjacent frames. Generally speaking, a small $MV(f_s, f_c)$ implies a stable head movement, while a large one indicates a considerable head pose change (Figure 4.7).

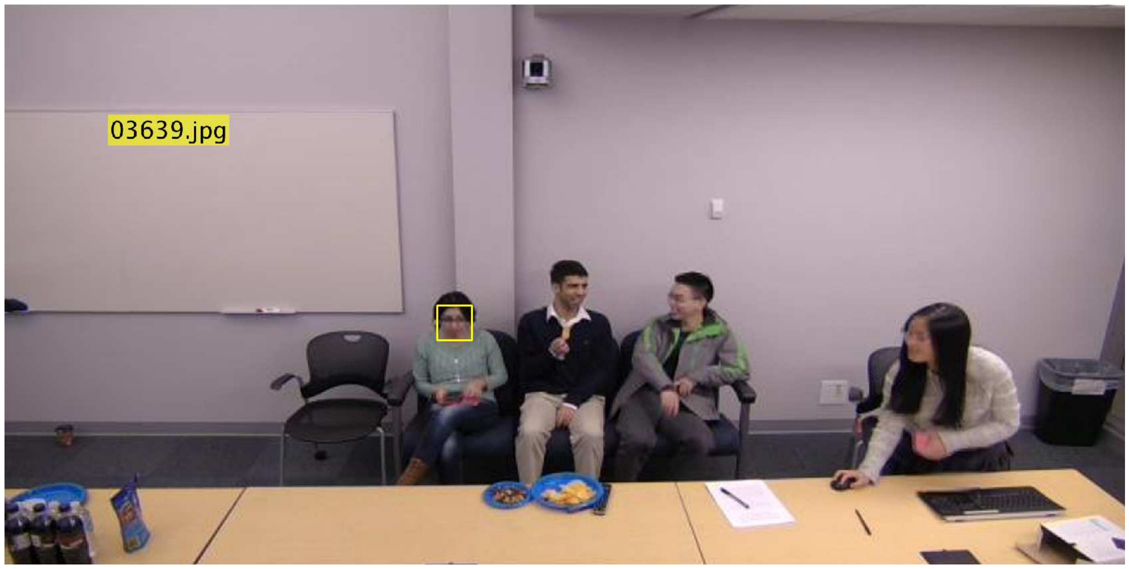
Finally, based on head pose change and $MV(f_s, f_c)$, a TV viewer's behavior is identified.

$$A\ viewer's\ behavior = \begin{cases} unfocused & \begin{array}{l} if\ |yaw_c - yaw_s| \geq T_1, or\ |tilt_c - tilt_s| \geq T_2, \\ or\ MV(f_s, f_c) \geq T_3 \end{array} \\ focused & \qquad otherwise \end{cases} \tag{4.8}$$

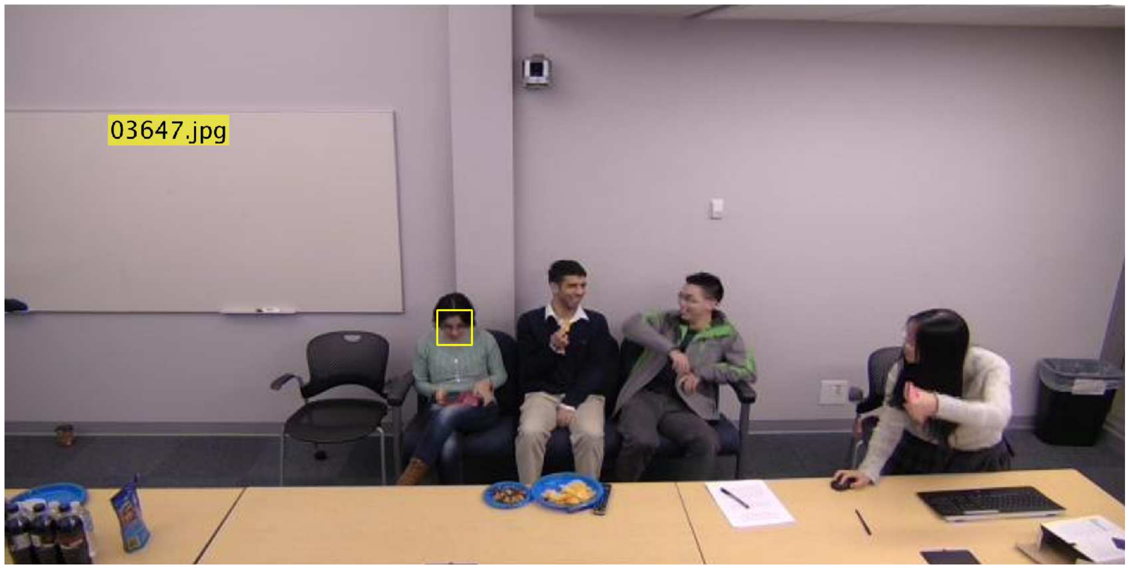
where yaw_s and $tilt_s$ represent yaw and tilt estimated from base frame f_s; yaw_c and $tilt_c$ represent yaw and tilt estimated from current frame f_c; T_1, T_2 and T_3 are threshold values.



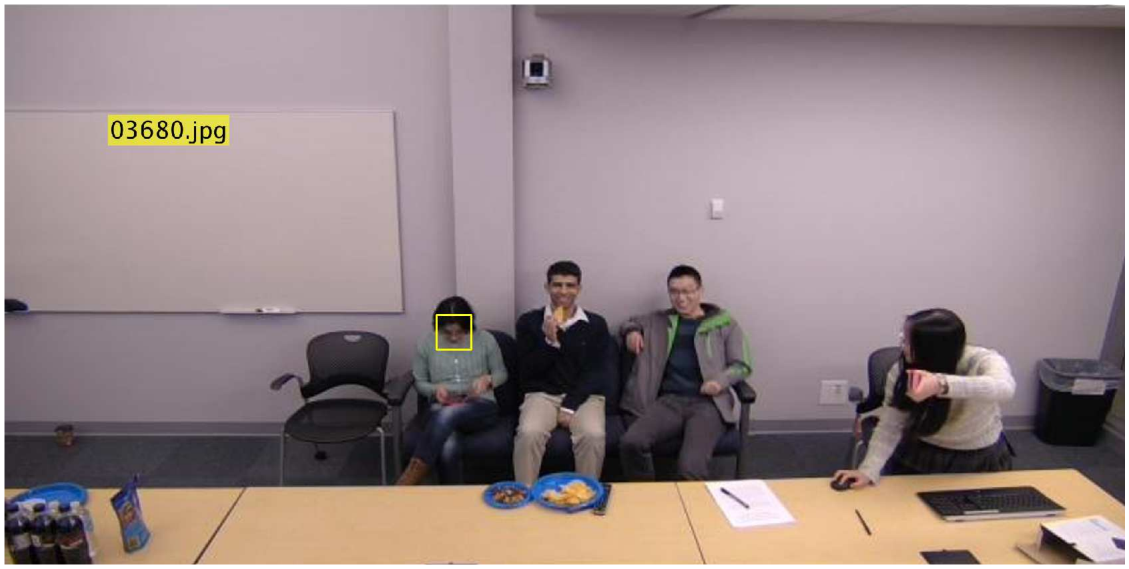
(a)



(b)



(c)



(d)

Figure 4.7: Face motion vector generated by head movement. (a) is the base frame, and the yellow rectangle indicates a detected face; in (b), (c) and (d), the yellow rectangles indicate face tracking results, with $MV(f_a, f_b)$ equal to (-1.0432, 1.1565), $MV(f_a, f_c)$ equal to (-1.3023, 9.8469), $MV(f_a, f_d)$ equal to (-2.3196, 19.0051). Significant head pose change happens in (c) and (d).

4.3 Summary

Our application is designed for analyzing TV viewers' behaviors and measuring their attention. Given the initial face detection results, the weighted KLT algorithm and consistency checking module generate stable face locations from the frame sequences and simultaneously output motion vectors to describe head movement. After that, head pose information is extracted by coarse-to-fine method. Finally, by selecting a base frame, TV viewer's behaviors are determined from relative head pose change and the face motion vector.

Chapter 5

Experimental Results

To demonstrate the superiority of coarse-to-fine head pose estimation method, we conduct experiments on CAS-PEAL-R1 face database [26] and Pointing'04 head pose image database [28], and then compare with state-of-the-art methods. After that, we test the application of analyzing TV viewers' behaviors, using eight videos from Boston University dataset [37] and two videos recorded by ourselves.

5.1 Performance of Coarse-to-fine estimation method

In this section, we give an introduction of comparison methods, followed by metrics for evaluating head pose estimation performance. After that, experimental results on two databases are presented.

5.1.1 Comparison Methods

Image descriptor and machine learning technique are critical elements for a head pose estimation method, therefore comparison methods are described from the two aspects.

Proposed in [29], Locally Adjusted Robust Regressor (LARR) makes use of concatenated raw pixels of an image as the image descriptor. Given a test image, two pre-trained SVRs separately generate yaw and tilt estimation. Afterwards, a set of pairs (*yaw, tilt*) closest to the initial estimation under Euclidean distance are selected from the original pose space. Based on the selected configurations, a SVM is trained to locally adjust the estimation for the test image.

The second comparison method, denoted as DenseSIFT method, was proposed in [30]. It forms a dense representation of an image, by extracting SIFT descriptors at image grid points. In DenseSIFT method, pose configurations are divided into several bins. A multi-class SVM is firstly employed to generate an initial estimation. Then a SVR is applied to get a refined result.

Li et al. [39] proposed a pixel-based method. For a face image, its corresponding binary image is generated by determining each pixel to be a skin pixel or not, and then divided into multiple blocks. Energy for each block is defined as the average pixel intensity, through which a Block Energy Map (BEM) is created. In this method, image descriptor is a feature vector concatenating energies for all blocks. To find a better learning technique, two regression method are tested, i.e., SVR and Gaussian Process Regression (GPR). The authors claimed GPR-based BEM yields better results.

Proposed in [51], the last comparison method makes use of Subclass Discriminant Analysis (SDA), thus it is called SDA-based method. Clustering process is first applied, and each group is further divided into several classes. Then the division serves as a preliminary step to learn a subspace based on SDA. By projecting original data into the subspace, classification is conducted by nearest neighbor searching.

The reason why we choose these four methods are listed as follows. Firstly, all of them except the third one share similar ideas as coarse-to-fine method in some sense. Both LARR and DenseSIFT methods obtain a refined estimation by searching a local neighborhood of initial results, and SDA-based method employs the same division idea as ours. Secondly, these papers are from recent years. The third one was published in 2014, and it employs GPR-based regression technique, which is popular and promising in recent studies.

5.1.2 Evaluation Metrics

To quantitatively evaluate the performance of different methods, we use the following metrics.

- Mean Absolute Error (MAE) [40]:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} e_i \tag{5.1}$$

$$= \frac{1}{N}\sum_{i=1}^{N} |g_i - r_i| \tag{5.2}$$

  where g_i is the ground truth for image i, and r_i is the estimated result; $|g_i - r_i|$ generates absolute error e_i for image i. N is the total number of testing images.

- Accuracy [17]:

$$Accuracy = \frac{n}{N} \times 100\% \tag{5.3}$$

  where n denotes the number of testing images for which absolute error is equal to 0, and N is the total number of testing images.

- Cumulative Error Distribution (CED) [71]:

$$CED(E) = \frac{N_{e \leq E}}{N} \tag{5.4}$$

  where $N_{e \leq E}$ is the number of testing images for which the absolute error e is no higher than E degrees; N is the total number of testing images.

Among three metrics, the first two are commonly used. Higher accuracy and lower mean absolute error indicate a better performance. Nevertheless, for some methods, such as regression based methods, the accuracy is usually 0, while the absolute error may be mostly within a small range. Therefore, to evaluate the performance from another point of view, cumulative error distribution is introduced. It can be viewed a function of E, and the output $CED(E)$ indicates the fraction of testing samples for which the absolute error is within the range $[0, E]$. In general, for a fixed error range E, a larger value of CED implies a better result.

5.1.3 CAS-PEAL-R1 Face Database

CAS-PEAL-R1 face database [26] consists of large amounts of grayscale images. Faces under different head poses are captured by a camera system consisting of nine digital cameras. Totally, images for 1042 people are contained in the database. For each person, there are totally 21 head poses, with seven yaw angles (ranging from $-45°$ to $45°$ with $15°$ interval) and three tilt angles ($-30°$, $0°$, and $30°$). Figure 5.1 shows a set of images for a person.

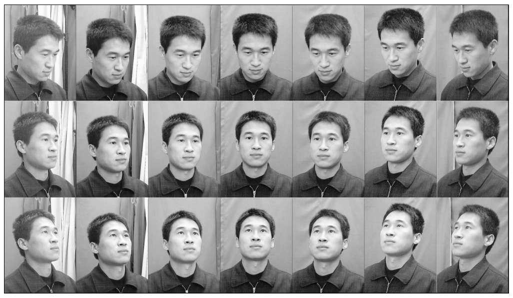


Figure 5.1: Example images from CAS-PEAL-R1 database. Each row corresponds to the same tilt angle and each column corresponds to the same yaw angle.

Considering the huge size of CAS-PEAL-R1 face database, we select a subset consisting of images of 90 people. Then we randomly choose images of 72 people as training instances, and the remaining images of 18 people as testing instances. This way of data division guarantees there is no overlap between training data and testing data, which is an important factor to test generality of a method.

To get face locations, we crop the face parts manually rather than making use of face detection algorithm, based on the following considerations. Firstly, all the images are in grayscale, thus they are not valid inputs for color-based skin detection method [7]; secondly, Viola-Jones algorithm [65] cannot detect faces under certain poses; thirdly, the manual cropping introduces little noise and generates standard face images. For images shown in Figure 5.1, cropped face parts are illustrated in Figure 5.2.

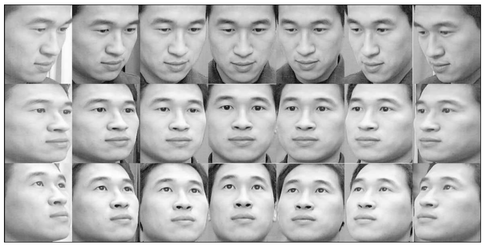


Figure 5.2: Cropped face parts of example images from CAS-PEAL-R1 database

Head pose estimation is conducted on the cropped face images. In coarse-to-fine estimation, yaw space is divided into three subspaces, i.e., $\{\{-45, -30\}, \{-15, 0, 15\}, \{30, 45\}\}$; three subspaces form the tilt space, i.e., $\{\{-30\}, \{0\}, \{30\}\}$. Through combined estimation of yaw and tilt, nine distinct group Ids are generated (Table 5.1). Meanwhile, each group is composed of several subclasses, for instance, two distinct within-group Ids are generated in the first group (Table 5.2). Finally, the label of an image corresponds to a vector (*group Id, within-group Id*), such as, vector $(1, 2)$ indicates the estimation for (*yaw,tilt*) is $(-30, -30)$.

Table 5.1: Group Ids generated in CAS-PEAL-R1 database

Yaw / Tilt	$\{-45, -30\}$	$\{-15, 0, 15\}$	$\{30, 45\}$
$\{-30\}$	1	2	3
$\{0\}$	4	5	6
$\{30\}$	7	8	9

Table 5.2: Within-group Ids of the first group

Yaw / Tilt	-45	-30
-30	1	2

Parameter selection in coarse-to-fine estimation method is given below. All the face images are resized to 64×64. In the coarse stage, the Gabor filter array is of size 5×8, and Gabor images are resized to 16×16; a random forest is built in coarse classifier training, with totally 300 decision trees. In the fine stage, to compute HOG descriptor, images are divided into cells of size 8×8, and Radial Basis Function (RBF) kernel is used in the SVM.

For the coarse-to-fine estimation method, the coarse estimation step achieves an accuracy of 94.44%. Among the 378 testing instances, the error introduced in the coarse estimation is shown in Figure 5.3. It can be seen that in each error case, the estimation result is a group with neighboring angles to the ground truth, indicating the coarse estimation guarantees the error within a small range.

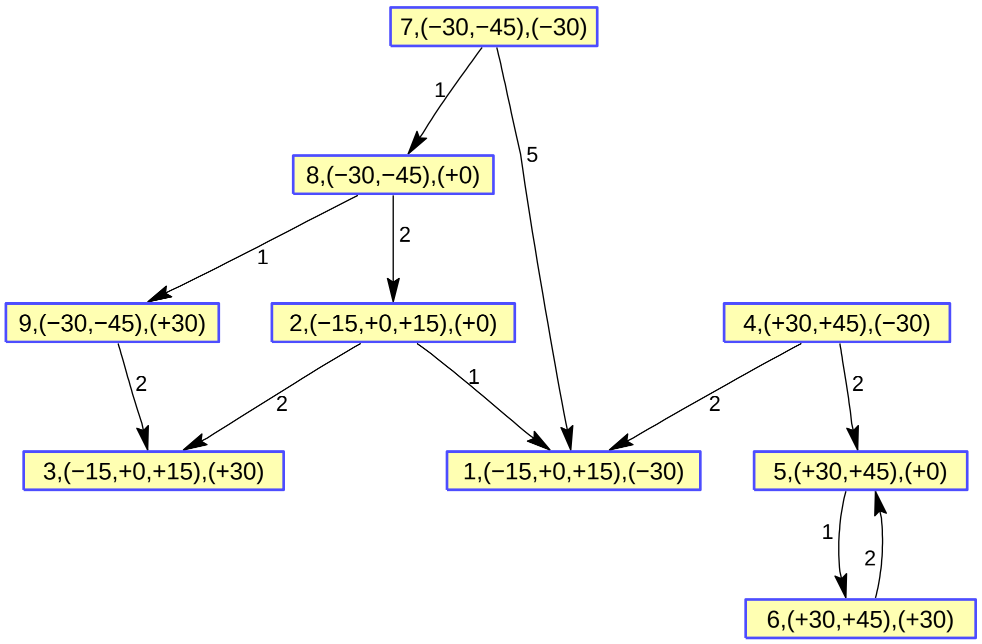


Figure 5.3: Error in the coarse estimation step for CAS-PEAL-R1 database. Each node is in the format of "*groupId*, (*yaw candidates*), (*tilt candidates*)"; each directed line connects the estimation to the ground truth, annotated with the number of occurrence.

The final result generated by coarse-to-fine method along with results by comparison methods are given below. Since head pose is determined by yaw and tilt, performance in terms of the sum of the two is the most convincing.

Table 5.3: Comparison of accuracy and MAE on CAS-PEAL-R1 database

Method	Accuracy (%)	MAE (°)
LARR	83.33	4.0079
DenseSIFT	0	11.8619
BEM+SVR [1]	N/A	N/A
BEM+GPR [2]	N/A	N/A
SDA	88.36	2.1429
Coarse-to-fine	**90.21**	**1.9048**

[1] BEM based method relies on color images as inputs, thus BEM+SVR is not applicable for CAS-PEAL-R1 database.
[2] For the same reason, BEM+GPR is not applicable for CAS-PEAL-R1 database.

Table 5.3 lists the accuracy and MAE in terms of the sum of yaw and tilt generated by different methods, and the best results are shown in bold. Table 5.3 shows that coarse-to-fine estimation method outperforms all the comparison methods. Besides, CED curves for different methods are illustrated in Figure 5.4. Coarse-to-fine method achieves the best performance, scoring 90.21% with 0° absolute error tolerance (i.e., accuracy), 98.00% with 15° tolerance, and 99.74% with 30° tolerance.

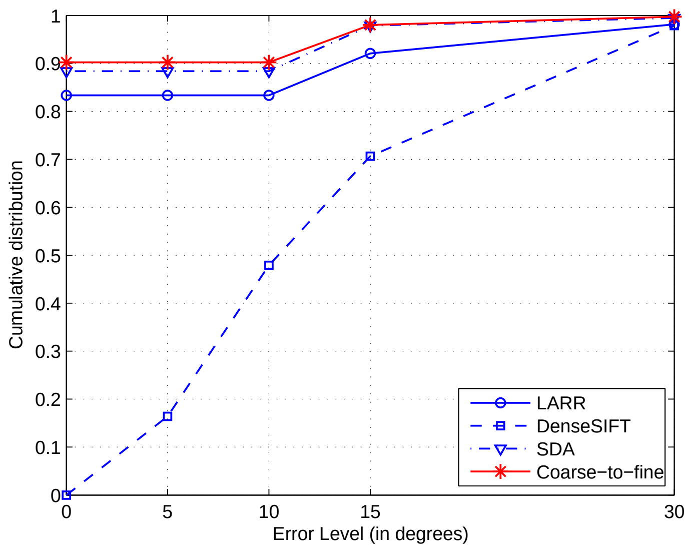
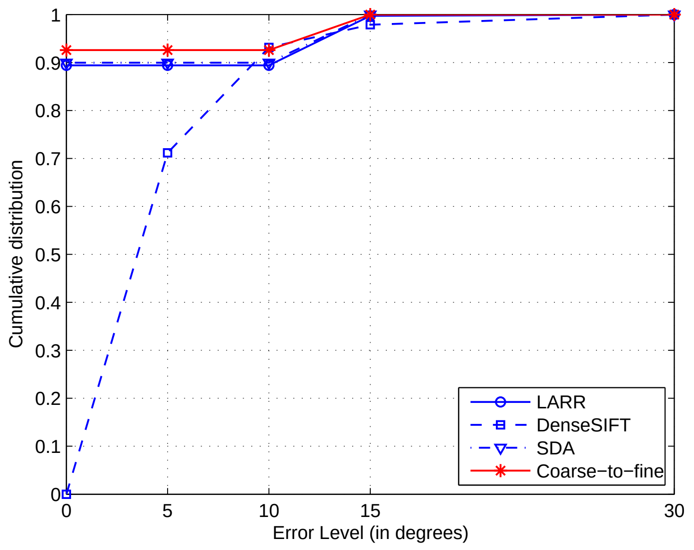


Figure 5.4: CED curves in terms of the sum of yaw and tilt on CAS-PEAL-R1 database

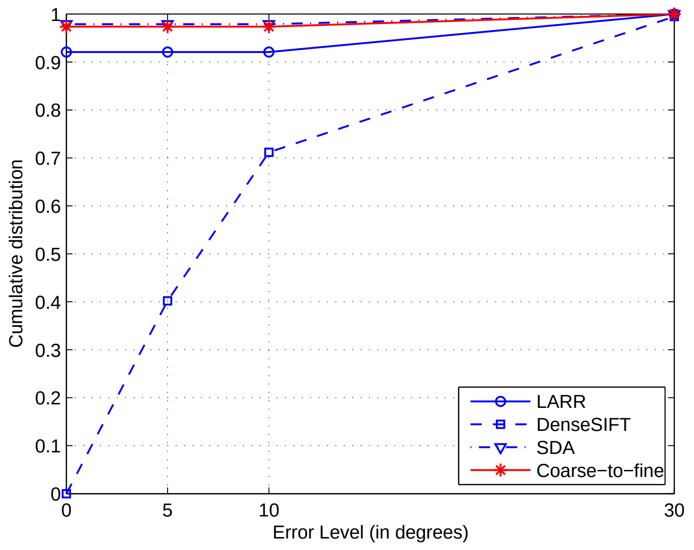
Furthermore, we compare the performance of different methods in terms of yaw and tilt separately. Table 5.4 lists the accuracy and MAE generated by different methods in yaw and tilt, and Figure 5.5 illustrates the CED curves. Table 5.4 and Figure 5.5 indicate that coarse-to-fine estimation method is always superior in evaluating yaw and tilt separately.

Table 5.4: Accuracy and MAE of yaw and tilt on CAS-PEAL-R1 database

Method	Accuracy (%)		MAE (°)	
	yaw	tilt	yaw	tilt
LARR	89.42	92.06	1.6270	2.3810
DenseSIFT	0	0	3.9873	7.8746
BEM+SVR	N/A	N/A	N/A	N/A
BEM+GPR	N/A	N/A	N/A	N/A
SDA	89.95	**97.88**	1.5079	**0.6379**
Coarse-to-fine	**92.59**	97.35	**1.1111**	0.7937



(a)



(b)

Figure 5.5: CED curves in terms of yaw (a) and tilt (b) on CAS-PEAL-R1 database

5.1.4 Pointing'04 Head Pose Image Database

Pointing'04 database [28] consists of face images of totally 15 people. In this database, people have various skin colors, and some of them wear glasses. For each person, there are 13 yaw angles $\{-90°, -75°, -60°, -45°, -30°, -15°, 0°, 15°, 30°, 45°, 60°, 75°, 90°\}$ and 9 tilt angles $\{-90°, -60°, -30°, -15°, 0°, 15°, 30°, 60°, 90°\}$. For images with tilt angles $-90°$ and $90°$, yaw angle is $0°$. As a result, the number of distinct head poses is $7 \cdot 13 + 2 \cdot 1 = 93$. Face part is invisible for images with tilt angles $-90°$ and $90°$, therefore, we focus on the remaining 91 poses.

For images in Pointing'04 database, color-based skin detection method [7] is employed to detect face locations. Figure 5.6 shows examples of face detection results.

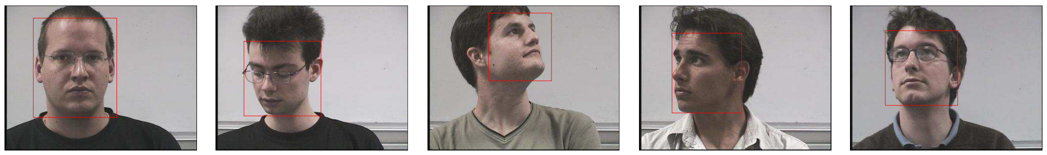

(a) $(0°, 0°)$ (b) $(15°, -60°)$ (c) $(-60°, 30°)$ (d) $(45°, 15°)$ (e) $(15°, 30°)$

Figure 5.6: Example images from pointing'04 database with red rectangles indicating the detected faces. Each image is annotated with (*yaw,tilt*).

Based on detected faces, head pose estimation is conducted using leave-one-out strategy. That is, images of 14 people act as training instances, while images of the remaining person act as testing instances. Each time we select one person as the testing person, thus totally 15 experiments are conducted. Performance of a head pose estimation method is evaluated from the average of all the experiments.

Implementation details of coarse-to-fine estimation method is as follows. Group Ids generated from the clustering process are listed in Table 5.5, and parameters in the coarse stage and fine stage are the same as in CAS-PEAL-R1 database.

Table 5.5: Group Ids in Pointing'04 database

Yaw / Tilt	$\{-90, -75\}$	$\{-60, -45, -30\}$	$\{-15, 0, 15\}$	$\{30, 45, 60\}$	$\{75, 90\}$
$\{-60, -30\}$	1	2	3	4	5
$\{-15, 0, 15\}$	6	7	8	9	10
$\{30, 60\}$	11	12	13	14	15

For coarse-to-fine estimation method, the average accuracy of coarse estimation step is 69.13%. Figure 5.7 shows the error introduced during the coarse estimation, in which 182 images of the 9*th* person act as testing instances. Similar to CAS-PEAL-R1 database, coarse estimation guarantees estimation results close to the ground truth.

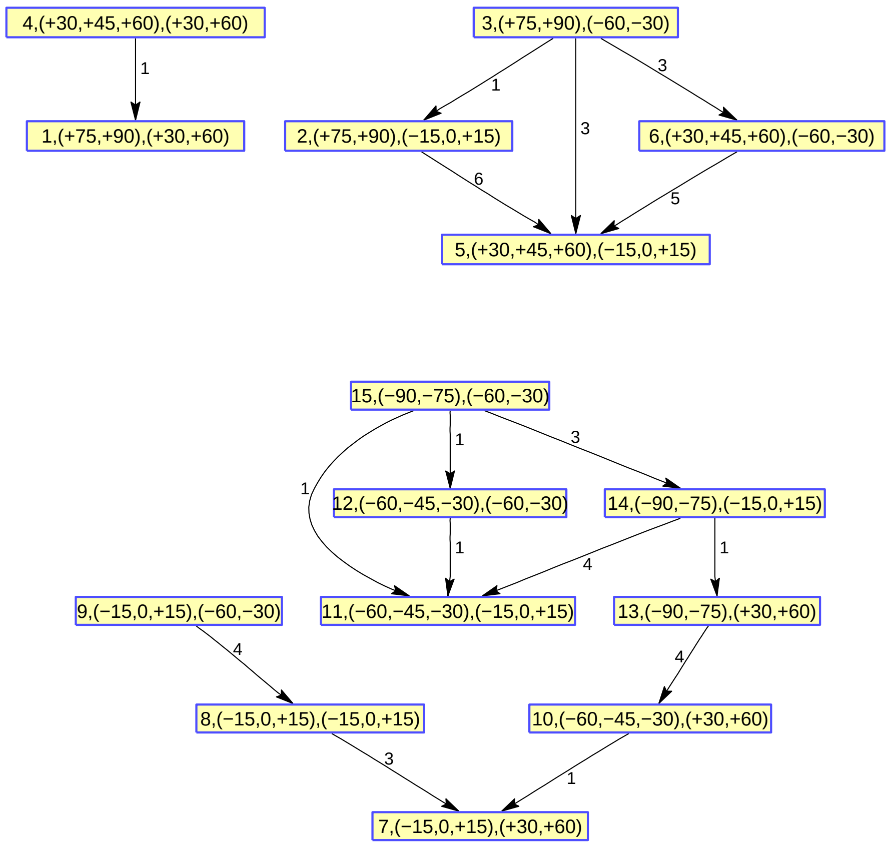


Figure 5.7: Error in the coarse estimation step for Pointing'04 database. Each node is in the format of "*groupId*, (*yaw candidates*), (*tilt candidates*)"; each directed line connects the estimation to the ground truth, annotated with the number of occurrence.

For different methods, accuracy and MAE in terms of the sum of yaw and tilt are listed in Table 5.6. It can be seen that coarse-to-fine estimation method works best.

Table 5.6: Comparison of accuracy and MAE on Pointing'04 database

Method	Accuracy (%)	MAE (°)
LARR	23.43	22.5300
DenseSIFT [1]	0	26.8837
BEM+SVR [2]	0	27.7223
BEM+GPR [3]	0	25.5657
SDA	27.52	17.8996
Coarse-to-fine	**33.32**	**15.8766**

[1] DenseSIFT method are regression based, thus generating zero accuracy.
[2] BEM+SVR relies on a SVR regression model, generating zero accuracy.
[3] BEM+GPR relies on a GPR regression model.

Figure 5.8 shows CED curves in terms of the sum of yaw and tilt. From Figure 5.8, coarse-to-fine estimation method scores 33.32% when no error is allowed, 74.93% within 15° error tolerance, and 92.01% within 30° tolerance, which means the majority of estimation results is within 30° error range.

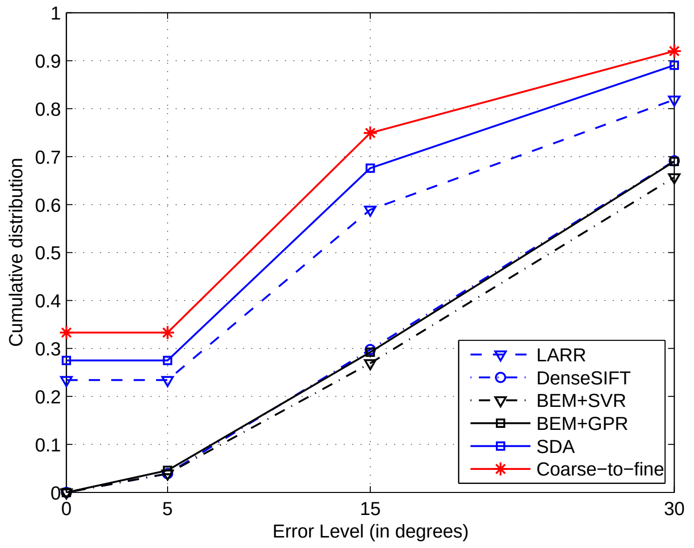


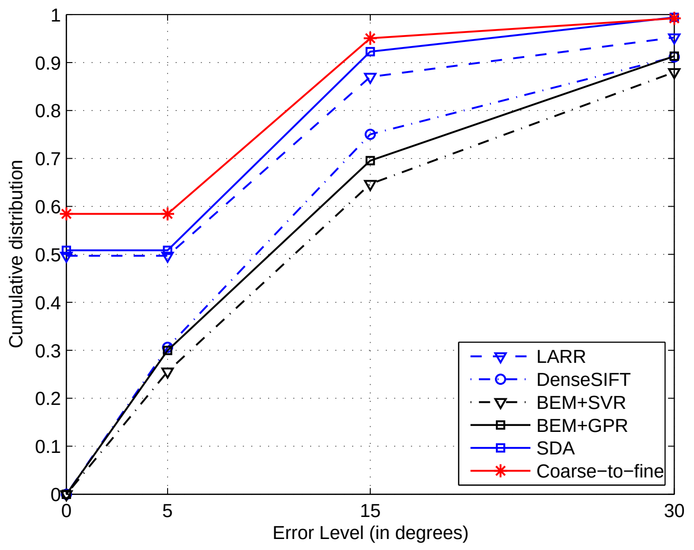
Figure 5.8: CED curves in terms of the sum of yaw and tilt on Pointing'04 database

The performance of different methods in estimating yaw and tilt separately is shown in Table 5.7 and Figure 5.9. It can be seen that coarse-to-fine estimation method outperforms the others in estimating yaw and tilt in a separate way.

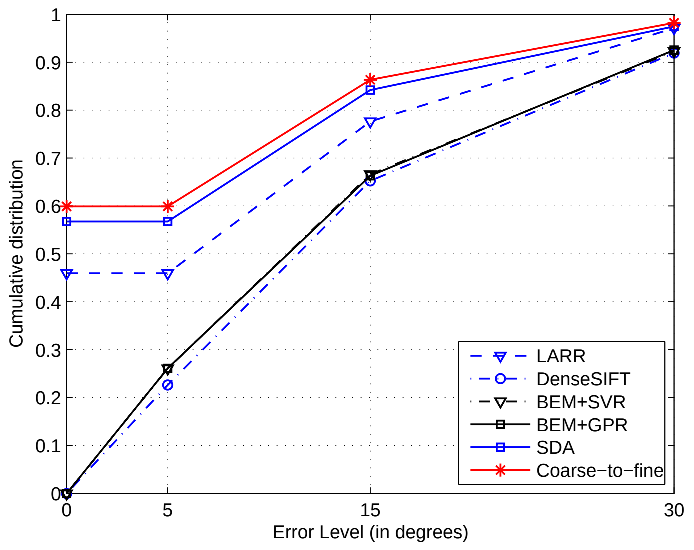
Table 5.7: Accuracy and MAE of yaw and tilt on Pointing'04 database

Method	Accuracy (%)		MAE (°)	
	yaw	tilt	yaw	tilt
LARR	49.70	45.95	10.6019	11.9281
DenseSIFT	0	0	12.8514	14.0323
BEM+SVR	0	0	14.8506	12.8717
BEM+GPR	0	0	12.7292	12.8366
SDA	50.85	56.74	8.6239	9.2757
Coarse-to-fine	**58.44**	**59.89**	**7.4925**	**8.3841**

Compared to CAS-PEAL-R1 face database, pointing'04 head pose image database contains fewer images. Besides, face detection on pointing'04 database introduces more noise. As a consequence, head pose estimation on Pointing'04 database is a more challenging task. In this sense, coarse-to-fine estimation method demonstrates its robustness.



(a)



(b)

Figure 5.9: CED curves in terms of yaw (a) and tilt (b) on Pointing'04 database

5.2 Performance of analyzing TV viewers' behaviors

5.2.1 Boston University Dataset

Boston university dataset [37] consists of 45 videos. Each video is digitized at 30 frames per second (FPS), consisting of 200 frames at a resolution of 320×240 (size of the face in each frame is around 90×90). In a video, only one person is monitored.

In our experiment, we select 8 videos as testing sequences. The first step is generating the ground truth, by manually labeling a TV viewer's behavior in each frame as focused or unfocused. After that, the viewer's behavior is analyzed by the pre-built application. In head pose estimation task, faces are resized to 64×64, which is the same as the size of images used to train the estimation method. Besides, resized faces also guarantee our method is robust to the distance between a person and the camera. Figure 5.10 shows some results of behavior analysis. Table 5.8 shows the accuracy for all the test sequences. Using a laptop computer with Intel(R) Core(TM) i5-3230M CPU @2.60GHz and 4.0GB memory, the average speed is 0.2386 seconds/frame.

Table 5.8: Accuracy for test sequences from Boston university dataset

Sequence	jam5	jam6	jim7	jim9	llm5	llm8	ssm5	ssm9
Accuracy (%)	97.5	89	88	92.5	93.5	98	84	85.5

5.2.2 Videos recorded by ourselves

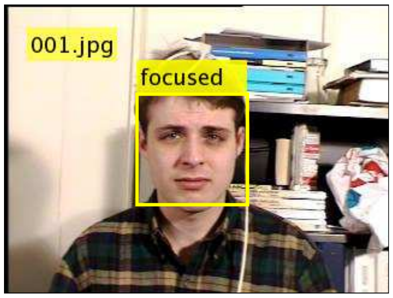
For the purpose of testing multi-person scenarios, two videos are recorded by ourselves. In each video, four people are watching a live hockey game displaying in a projector. Besides, people can move freely, check their smart phones and talk with their neighbors. By manually annotating people's behaviors to get the ground truth, representative video clips are extracted and tested using our application. Figures 5.11 and 5.12 show some results for the two video recordings. Table 5.10 lists the accuracy for the two videos.

Table 5.9: Properties of the two videos recorded on our own

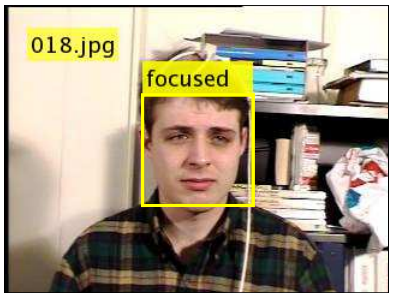
Sequence	FPS	No.of frames	Resolution	Rough size of a face
Video1	29	27949	720×486	20×20
Video2	6	19755	640×480	35×35

Table 5.10: Accuracy for the two videos recorded on our own

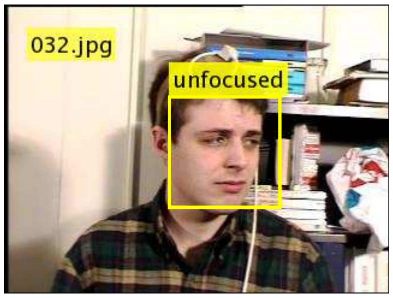
Sequence	Video1	Video2
Accuracy (%)	63.33	74.5



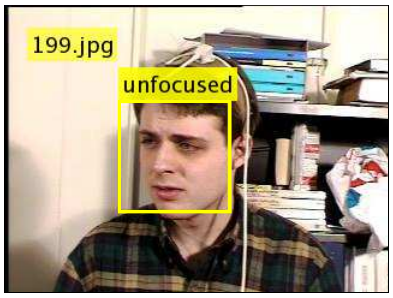
(a)



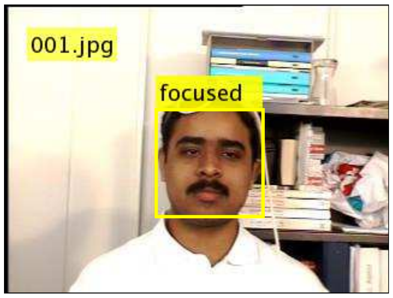
(b)



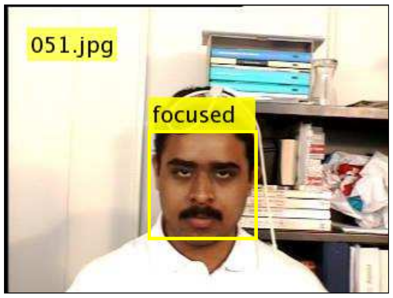
(c)



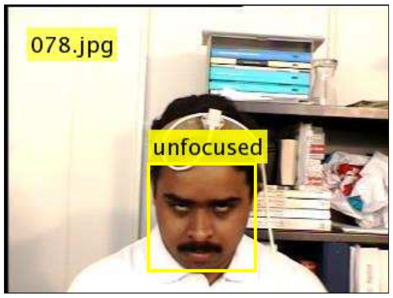
(d)



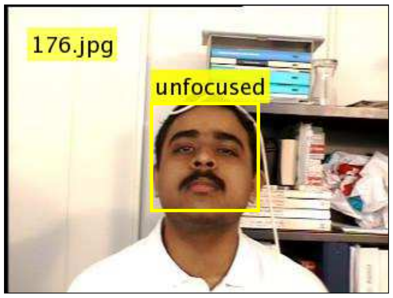
(e)



(f)

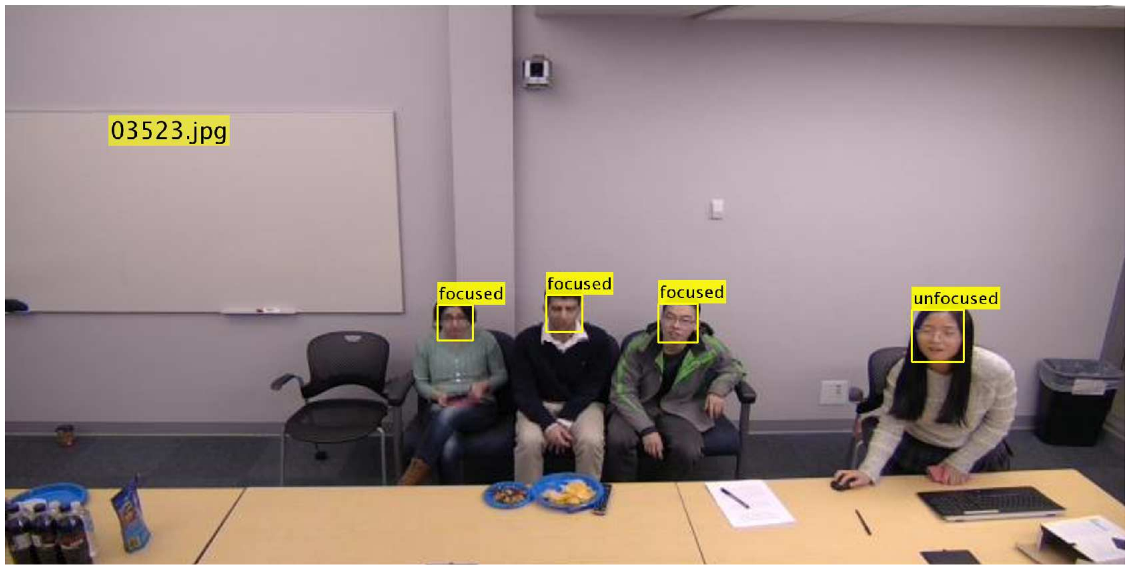


(g)

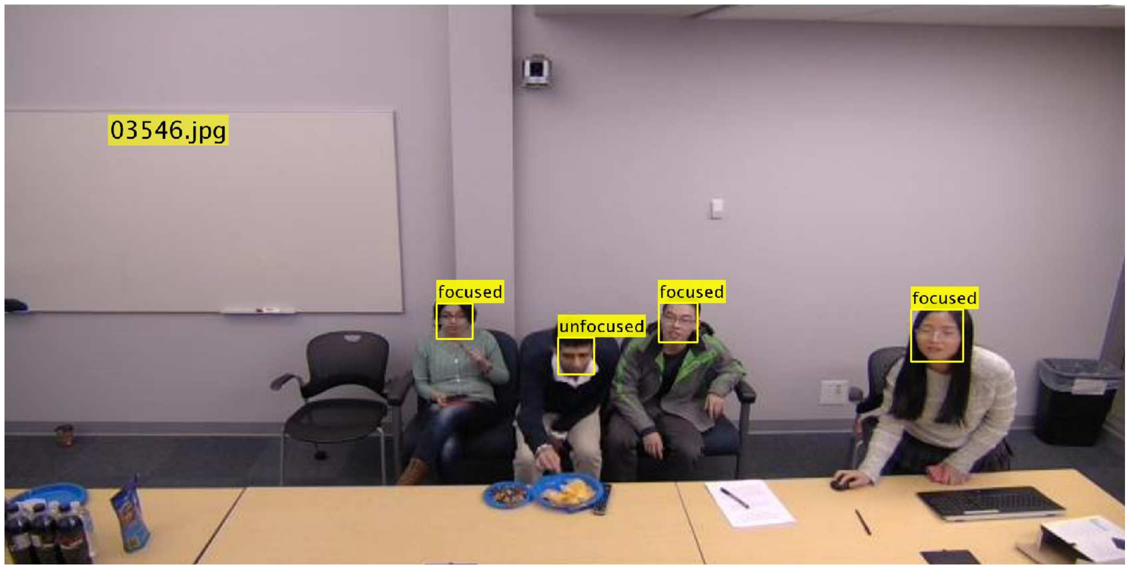


(h)

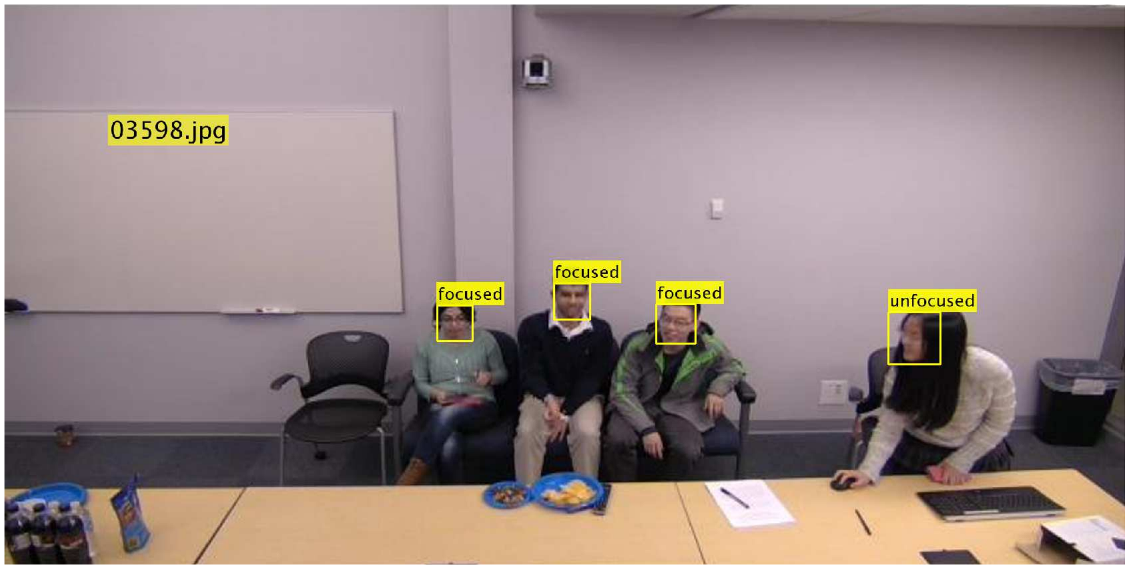
Figure 5.10: Results of TV viewers' behavior analysis on Boston university dataset. The first four frames are from sequence 'jim9', and (a) is the base frame; the last four frames are from 'ssm5', with (e) as the base frame.


03523.jpg
focused
focused
focused
unfocused

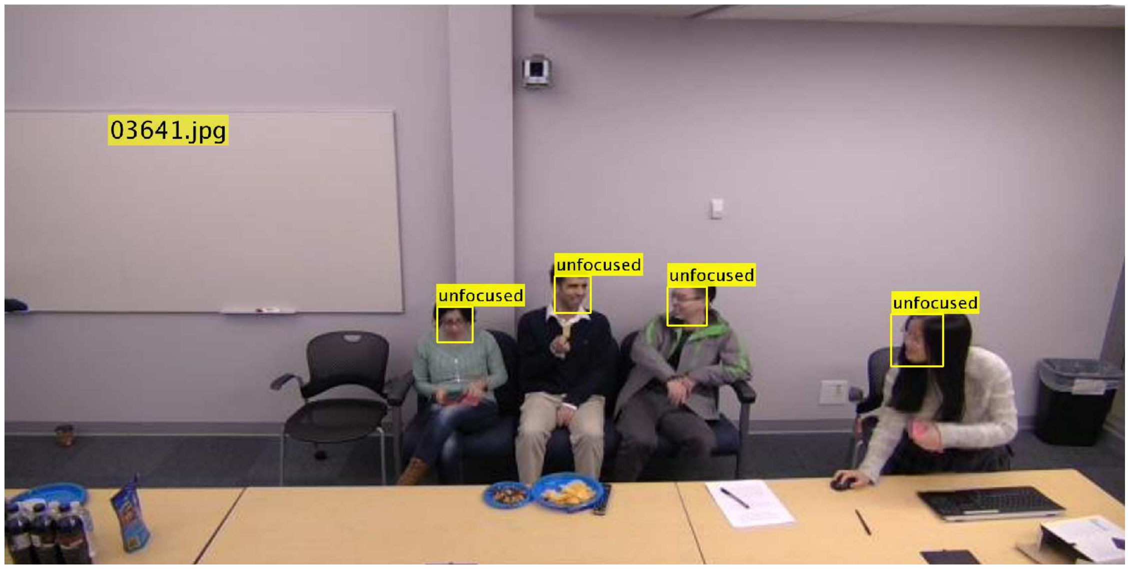
(a)


03546.jpg
focused
unfocused
focused
focused

(b)

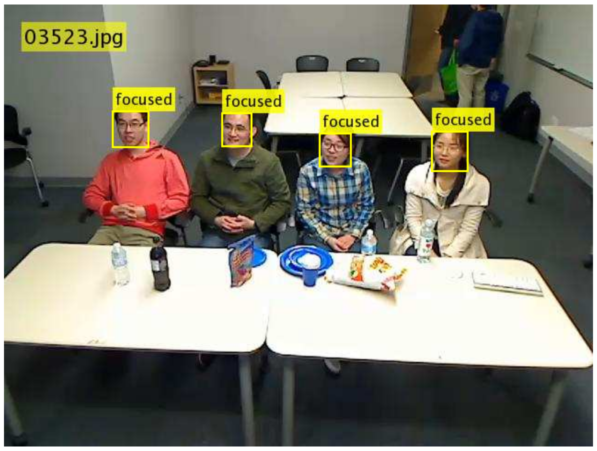


(c)

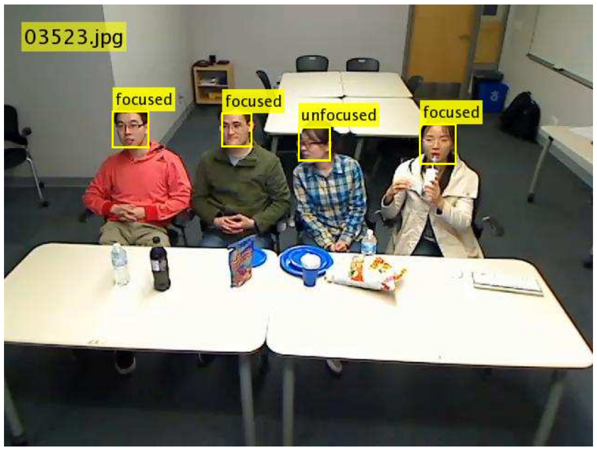


(d)

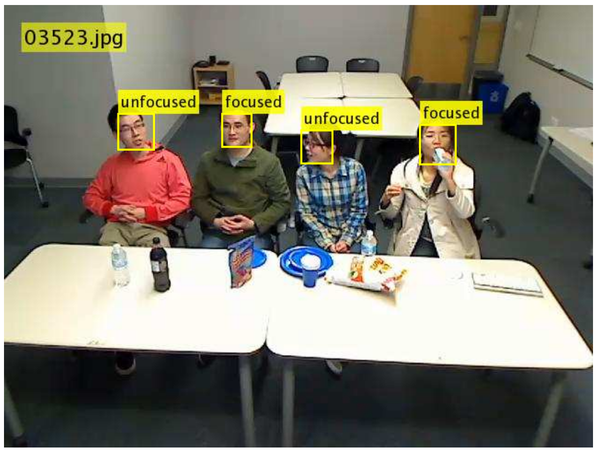
Figure 5.11: Results of TV viewers' behavior analysis on Video1 (New People coming into the view are detected by face detection, while when people leave the view, feature points used to track the people would be no longer tracked)



(a)



(b)



(c)

Figure 5.12: Results of TV viewers' behavior analysis on Video2

Chapter 6

Conclusion and Future Work

In this thesis, we present a characterization study of existing head pose estimation methods and propose a coarse-to-fine head pose estimation method. In addition, we apply our method to analyze TV viewers' behaviors.

We first investigate various inputs required by different estimation methods and analyze the feasibility of different set-ups. We then focus on estimation methods with wide applicability. Based on the strategy of how to extract information face images, head pose estimation methods are divided into three categories. By analyzing the pros and cons of methods of three categories, we propose a coarse-to-fine estimation method by extracting information from the entire face image.

A two-step estimation mechanism is proposed in our method. Through clustering process, the original pose space can be decomposed in a hierarchical structure with three layers. Correspondingly, two estimation steps can be viewed as the process of identifying paths between neighboring layers.

Additionally, ROI detection module is proposed in our method, based on the finding that image points implicitly take on different degrees of importance in head pose estimation task. By exploring importance degree of image points, ROI is detected by incorporating the most important points within an image.

Finally, we build an application of interpreting TV viewers' behaviors by integrating a sequence of tasks. In face tracking, a weighting function of feature points detected by KLT and consistency checking module generate reliable and stable tracking results. To deal with continuous head pose change in a video sequence, we identify a viewer's behavior from head pose change measurement and face motion vector.

Current problems and suggestions for future work are listed below.

1. Coarse-to-fine estimation method relies on classification based machine learning technique. To enhance video-based applications, a regression model on continuous pose space is needed.

2. In coarse estimation, Gabor filter is employed to extract the feature vectors of input images. The orientation of a Gabor filter is defined within X-Y plane (based on Figure 2.1). To better reveal image features in terms of the yaw and tilt, a 3D filter projected onto X-Z and Y-Z planes would be a better choice.

3. In the application of analyzing TV viewers' behaviors, face detection, face tracking and head pose estimation are isolated tasks. To achieve a better performance in a more efficient way, one solution is to build a unified model for the three tasks.

4. In our application, behavior of each TV viewer is analyzed independently. In multi-person scenarios, behaviors of multiple viewers can be interpreted jointly, in order to understand the interaction between them.

Bibliography

[1] Sileye O Ba and Jean-Marc Odobez. Recognizing visual focus of attention from head pose in natural meetings. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(1):16–33, 2009.

[2] Ben Benfold and Ian Reid. Guiding visual surveillance by tracking human attention. In *BMVC*, pages 1–11, 2009.

[3] Stan Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 232–237. IEEE, 1998.

[4] Jurjen Caarls. *Pose estimation for mobile devices and augmented reality.* TU Delft, Delft University of Technology, 2009.

[5] Qin Cai, Alamelu Sankaranarayanan, Q Zhang, Zhengyou Zhang, and Zicheng Liu. Real time head pose tracking from multiple cameras with a generic model. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 25–32. IEEE, 2010.

[6] Isarun Chamveha, Yusuke Sugano, Daisuke Sugimura, Teera Siriteerakul, Takahiro Okabe, Yoichi Sato, and Akihiro Sugimoto. Appearance-based head pose estimation with scene-specific adaptation. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1713–1720. IEEE, 2011.

[7] Ciarán O Conaire, Noel E O'Connor, and Alan F Smeaton. Detector adaptation by maximising agreement between independent data sources. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–6. IEEE, 2007.

[8] Timothy F Cootes, Gareth J Edwards, and Christopher J Taylor. Active appearance models. In *Computer VisionâĂŤECCVâĂŹ98*, pages 484–498. Springer, 1998.

[9] Timothy F Cootes, Christopher J Taylor, David H Cooper, and Jim Graham. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

[10] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[11] Matthias Dantone, Juergen Gall, Gabriele Fanelli, and Luc Van Gool. Real-time facial feature detection using conditional regression forests. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2578–2585. IEEE, 2012.

[12] John G Daugman. Two-dimensional spectral analysis of cortical receptive field profiles. *Vision research*, 20(10):847–856, 1980.

[13] John G Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *JOSA A*, 2(7):1160–1169, 1985.

[14] Oscar Déniz, Gloria Bueno, Jesús Salido, and Fernando De la Torre. Face recognition using histograms of oriented gradients. *Pattern Recognition Letters*, 32(12):1598–1603, 2011.

[15] Ramón Díaz-Uriarte and Sara Alvarez De Andres. Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, 7(1):3, 2006.

[16] Thomas G Dietterich. Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer, 2000.

[17] Ligeng Dong, Linmi Tao, and Guangyou Xu. Head pose estimation using covariance of oriented gradients. In *ICASSP*, pages 1470–1473, 2010.

[18] Ligeng Dong, Linmi Tao, Guangyou Xu, and Patrick Oliver. A study of two image representations for head pose estimation. In *Image and Graphics, 2009. ICIG'09. Fifth International Conference on*, pages 963–968. IEEE, 2009.

[19] Reza Ebrahimzadeh and Mahdi Jampour. Efficient handwritten digit recognition based on histogram of oriented gradients and svm. *International Journal of Computer Applications*, 104(9), 2014.

[20] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[21] Gabriele Fanelli, Thibaut Weise, Juergen Gall, and Luc Van Gool. Real time head pose estimation from consumer depth cameras. In *Pattern Recognition*, pages 101–110. Springer, 2011.

[22] Robert Fisher, Simon Perkins, Ashley Walker, and Erik Wolfart. Connected component labeling. *website: http://homepages. inf. ed. ac. uk/rbf/HIPR2/label. htm*, 2003.

[23] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[24] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[25] Dennis Gabor. Theory of communication. part 1: The analysis of information. *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering*, 93(26):429–441, 1946.

[26] Wen Gao, Bo Cao, Shiguang Shan, Xilin Chen, Delong Zhou, Xiaohua Zhang, and Debin Zhao. The cas-peal large-scale chinese face database and baseline evaluations. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 38(1):149–161, 2008.

[27] Robin Genuer, Jean-Michel Poggi, and Christine Tuleau-Malot. Variable selection using random forests. *Pattern Recognition Letters*, 31(14):2225–2236, 2010.

[28] Nicolas Gourier, Daniela Hall, and James L Crowley. Estimating face orientation from robust detection of salient facial structures. In *FG Net Workshop on Visual Observation of Deictic Gestures*, pages 1–9. FGnet (IST–2000–26434) Cambridge, UK, 2004.

[29] Guodong Guo, Yun Fu, Charles R Dyer, and Thomas S Huang. Head pose estimation: Classification or regression? In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008.

[30] Huy Tho Ho and Rama Chellappa. Automatic head pose estimation using randomly projected dense sift descriptors. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 153–156. IEEE, 2012.

[31] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, 2002.

[32] Kohsia S Huang and Mohan M Trivedi. Robust real-time detection, tracking, and pose estimation of faces in video streams. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 965–968. IEEE, 2004.

[33] Takahiro Ishikawa. Passive driver gaze tracking with active appearance models. 2004.

[34] Min Jiang, Lin Deng, Lei Zhang, J Tang, and Chan Fan. Head pose estimation based on active shape model and relevant vector machine. In *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pages 1035–1038. IEEE, 2012.

[35] Hyunduk Kim, Sang-Heon Lee, Myoung-Kyu Sohn, and Dong-Ju Kim. Illumination invariant head pose estimation using random forests classifier and binary pattern run length matrix. *Human-centric Computing and Information Sciences*, 4(1):1–12, 2014.

[36] Farid Abedan Kondori, Sh Yousefi, and Haibo Li. Direct three-dimensional head pose estimation from kinect-type sensors. *Electronics Letters*, 50(4):268–270, 2014.

[37] Marco La Cascia and Stan Sclaroff. Fast, reliable head tracking under varying illumination. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 1. IEEE, 1999.

[38] Stephen RH Langton, Helen Honeyman, and Emma Tessler. The influence of head contour and nose angle on the perception of eye-gaze direction. *Perception & psychophysics*, 66(5):752–771, 2004.

[39] Wei Li, Yan Huang, and Jingliang Peng. Automatic and robust head pose estimation by block energy map. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 3357–3361. IEEE, 2014.

[40] Yali Li, Shengjin Wang, and Xiaoqing Ding. Person-independent head pose estimation based on random forest regression. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 1521–1524. IEEE, 2010.

[41] Bingpeng Ma, Shiguang Shan, Xilin Chen, and Wen Gao. Head yaw estimation from asymmetry of facial appearance. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 38(6):1501–1512, 2008.

[42] Bingpeng Ma and Tianjiang Wang. Head pose estimation using sparse representation. In *Computer Engineering and Applications (ICCEA), 2010 Second International Conference on*, volume 2, pages 389–392. IEEE, 2010.

[43] James G MacKinnon. Bootstrap hypothesis testing. *Handbook of Computational Econometrics*, pages 183–213, 2009.

[44] Manas K Mandal and Avinash Awasthi. Understanding facial expressions in communication. 2015.

[45] Julio C Mateo, Javier San Agustin, and John Paulin Hansen. Gaze beats mouse: hands-free selection by combining gaze and emg. In *CHI'08 extended abstracts on Human factors in computing systems*, pages 3039–3044. ACM, 2008.

[46] Rodrigo Minetto, Nicolas Thome, Matthieu Cord, Neucimar J Leite, and Jorge Stolfi. T-hog: An effective gradient-based descriptor for single line text regions. *Pattern recognition*, 46(3):1078–1090, 2013.

[47] Rafael Muñoz-Salinas, Enrique Yeguas-Bolivar, Alessandro Saffiotti, and R Medina-Carnicer. Multi-camera head pose estimation. *Machine Vision and Applications*, 23(3):479–490, 2012.

[48] Erik Murphy-Chutorian, Anup Doshi, and Mohan Manubhai Trivedi. Head pose estimation for driver assistance systems: A robust algorithm and experimental evaluation. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 709–714. IEEE, 2007.

[49] Erik Murphy-Chutorian and Mohan Manubhai Trivedi. Head pose estimation in computer vision: A survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(4):607–626, 2009.

[50] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):971–987, 2002.

[51] HB Oost. Sda-based discrete head pose estimation. 2009.

[52] Carlos Orrite, Andrés Gañán, and Grégory Rogez. Hog-based decision tree for facial expression classification. In *Pattern Recognition and Image Analysis*, pages 176–183. Springer, 2009.

[53] William H Press, Saul A Teukolsky, William T Vetterling, and BP Flannery. Section 16.5. support vector machines. *Numerical Recipes: The Art of Scientific Computing*, 2007.

[54] Jianfeng Ren, Mohammad Rahman, Nasser Kehtarnavaz, and Leonardo Estevez. Real-time head pose estimation on mobile platforms. *Journal of Systemics, Cybernetics and Informatics*, 8(3):56–62, 2010.

[55] Ravikrishna Ruddarraju, Antonio Haro, and Irfan Essa. Fast multiple camera head pose tracking. In *Vision Interface*, volume 2. Citeseer, 2003.

[56] Michael Sapienza and Kenneth P Camilleri. Fasthpe: A recipe for quick head pose estimation. 2014.

[57] Edgar Seemann, Kai Nickel, and Rainer Stiefelhagen. Head pose estimation using stereo vision for human-robot interaction. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 626–631. IEEE, 2004.

[58] Jianbo Shi and Carlo Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[59] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[60] Kevin Smith, Sileye O Ba, Jean-Marc Odobez, and Daniel Gatica-Perez. Tracking the visual focus of attention for a varying number of wandering people. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(7):1212–1229, 2008.

[61] Rainer Stiefelhagen. Estimating head pose with neural networks-results on the pointing04 icpr workshop evaluation data. In *PointingâĂŹ04 ICPR Workshop of the Int. Conf. on Pattern Recognition*, 2004.

[62] Rainer Stiefelhagen, Michael Finke, Jie Yang, and Alex Waibel. From gaze to focus of attention. In *Visual Information and Information Systems*, pages 765–772. Springer, 1999.

[63] Carlo Tomasi and Takeo Kanade. *Detection and tracking of point features.* School of Computer Science, Carnegie Mellon Univ. Pittsburgh, 1991.

[64] Teodora Vatahska, Maren Bennewitz, and Sven Behnke. Feature-based head pose estimation from images. In *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, pages 330–335. IEEE, 2007.

[65] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[66] Michael Voit, Kai Nickel, and Rainer Stiefelhagen. Multi-view head pose estimation using neural networks. In *Computer and Robot Vision, 2005. Proceedings. The 2nd Canadian Conference on*, pages 347–352. IEEE, 2005.

[67] Dirk W Wagener and Ben Herbst. Face tracking: An implementation of the kanade-lucas-tomasi tracking algorithm. *PRASA, South Africa*, 2001.

[68] Dimitri J Walger, Toby P Breckon, Anna Gaszczak, and Thomas Popham. A comparison of features for regression-based driver head pose estimation under varying illumination conditions. In *Computational Intelligence for Multimedia Understanding (IWCIM), 2014 International Workshop on*, pages 1–5. IEEE, 2014.

[69] Yucheng Wei, Ludovic Fradet, and Tieniu Tan. Head pose estimation using gabor eigenspace modeling. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–281. IEEE, 2002.

[70] Zhenqiu Zhang, Yuxiao Hu, Ming Liu, and Thomas Huang. Head pose estimation in seminar room using multi view face detectors. In *Multimodal Technologies for Perception of Humans*, pages 299–304. Springer, 2007.

[71] Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2879–2886. IEEE, 2012.

Appendix A

Viola-Jones Face Detection

A.1 Haar-like Features

For an image, its Haar-like features are intensity difference between rectangular regions that have similar shape with Haar wavelet. In Viola-Jones face detection algorithm [65], there are three kinds of Haar-like features: two-rectangle feature, three-rectangle feature and four-rectangle feature, illustrated in Figure A.1.

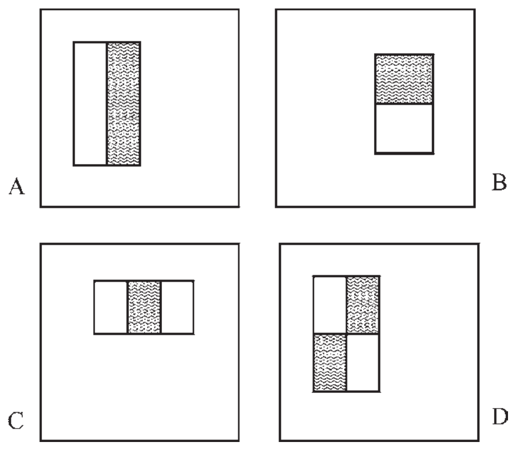


Figure A.1: Rectangular regions within an image for Haar-like features [65]

The rectangle feature for an image is calculated by

$$f = \sum_{i \in W} I(i) - \sum_{j \in G} I(j) \tag{A.1}$$

where I is the original 2D image; i and j denote image pixels; W represents the while rectangle regions, and G represents the gray rectangle regions.

A.2 Integral Image

To reduce the computational cost of calculating Haar-like features, the integral image is introduced:

$$ii(x, y) = \sum_{x^* \leq x,\, y^* \leq y} I(x^*, y^*) \tag{A.2}$$

where I is the original image; (x, y) and (x^*, y^*) represent pixel coordinates; ii is the integral image.

Equation A.2 indicates the intensity of the integral image ii at (x, y) is the sum of pixels in upper left regions from the original image. Based on the integral image, the sum of pixels within rectangles of the original image can be rapidly computed.

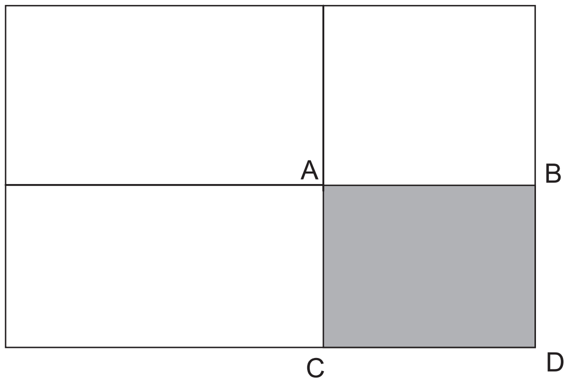


Figure A.2: Calculating the sum of pixels within a rectangle by the integral image. The sum of pixels within the gray rectangle can be computed by $ii(A) + ii(D) - ii(B) - ii(C)$, where ii is the integral image, and A, B, C, D are four vertices of the gray rectangle.

A.3 AdaBoost

Given Haar-like features and a training set, AdaBoost is used to build classifiers.

Proposed by Freund and Schapire [23], AdaBoost is a typical boosting method. The weighted training set is employed in AdaBoost, and the weights keep changing during the training process. Initially, uniform weights are assigned to instances in the training set. In each iteration, a classifier is learned by the training set in current state. For the trained classifier, some instances are well classified while some are not. To improve accuracy of the classifier successively, in next iteration, more attention should be paid to the misclassified instances. To this end, the training set is updated by assigning more weight to misclassified instances and less weight to correctly classified ones. Finally, a boosted classifier is generated by combining classifiers trained in different iterations, and the weight of each classifier is proportional to its classification accuracy [16].

In conclusion, AdaBoost builds a strong classifier from a sequence of weak classifiers. The general procedure of AdaBoost is summarized below.

Algorithm 2 A general procedure for AdaBoost

```
Denote $T_r$ as the training set and $k$ as the number of iterations.
Training:
  Initialize uniform weight for training instances from $T_r$.
  for $i = 1$ to $k$ do
    train classifier $C_i$ by weighted set $T_r$;
    make classifications on training data ;
    update weights for training instances.
  end for
  return A boosted classifier
```

Appendix B

Connected-component Labeling

In binary image analysis, the goal of connected-component labeling is to generate a labeled image, in which each pixel is assigned with a unique label and pixels with the same label belong to a connected region [22].

Within a binary image, labeling the connected component of a given pixel is based on checking the values of its neighbors. Usually, 8-connected pixels are used to represent the neighbors of a pixel. For a pixel at (x, y), 8 pixels at coordinates $(x \pm 1, y)$, $(x, y \pm 1)$, $(x \pm 1, y \pm 1)$ and $(x \pm 1, y \mp 1)$ are its neighbors. Connected components are neighbors that share the same value with the pixel at (x, y).

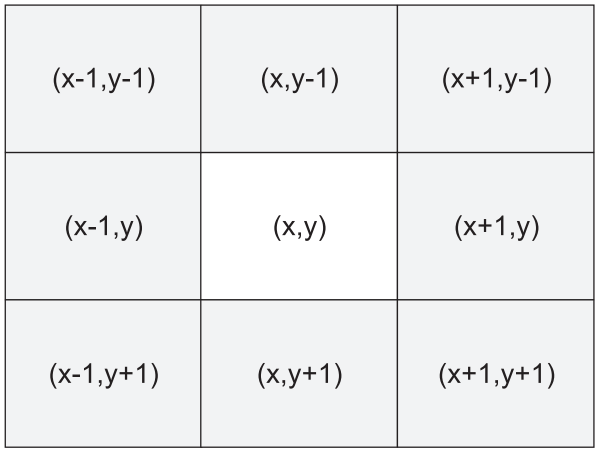


Figure B.1: 8-connected neighbor pixels. For the center pixel (x, y), its neighbors are represented by the gray rectangles, from the horizontal, vertical and diagonal directions.

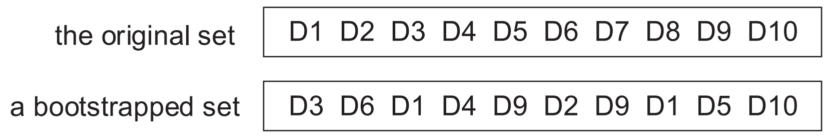
Appendix C

Bootstrap Sampling

In averaging methods, each base classifier relies on a training set, generated from the original training set through bootstrap sampling [20].

The principle of bootstrap sampling is described below. Denote T as the original training set of size N, and T^* as the training set for a base classifier. In averaging methods, T^* is of the same size as T. Bootstrap sampling is the random sampling with replacement. That is, each instance in T^* is drawn from the whole set T randomly, and a set of N instances requires N independent sampling processes.

Through bootstrap sampling, some instances from original set T may appear in T^* more than once, while some instances belonging to T may be absent in T^*. A simple example is illustrated in Figure C.1.



the original set
D1 D2 D3 D4 D5 D6 D7 D8 D9 D10
a bootstrapped set
D3 D6 D1 D4 D9 D2 D9 D1 D5 D10

Figure C.1: An example of bootstrap sampling result, with D_i representing the ith instance, and $i = 1, ..., 10$.

It is easy to understand that each bootstrapped set T^* is correlated with the original set T to some extent. MacKinnon gives a mathematical proof in [43]. For T and T^* of size N, in each sampling, the probability of ignoring a specific instance is $1 - 1/N$. Since each sampling process is independent, the probability of ignoring an instance for N times is $(1 - 1/N)^N$. As $N \to \infty$, $(1 - 1/N)^N \to 1/e \approx 0.3679$, indicating that the probability of missing a specific instance in T^* is 0.3679. In turn, the probability of its presence in T^* is 0.6321. In other words, a bootstrapped set T^* contains 63.21% of the original training instances.

Appendix D

Consistency checking

D.1 LBP

LBP [50] is a texture descriptor extracted from gray images. By measuring the difference between a pixel and its neighborhood, LBP is defined by

$$LBP_{P,R}(x_c, y_c) = \sum\nolimits_{p=0}^{P-1} s(g_p - g_c)2^p \tag{D.1}$$

where (x_c, y_c) is the coordinate of a pixel with gray value g_c; totally P pixels are spaced on the circle with center (x_c, y_c) and radius R, and g_p ($p \in [1, P]$) denotes the gray value. The function $s(x)$ is given by

$$s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{D.2}$$

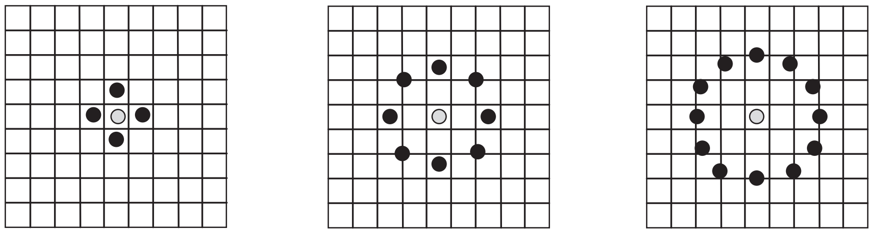


Figure D.1: Different sets of P and R values in LBP. (a) $P = 4, R = 1$; (b) $P = 8, R = 2$; (c) $P = 12, R = 2.5$.

D.2 Flowchart of consistency checking

Through consistency checking, a tracking result is determined to be valid or invalid. Denote c_1, c_2 and c_3 as the three criteria presented in Chapter 4.1.3. Figure D.2 illustrates the flowchart of consistency checking.

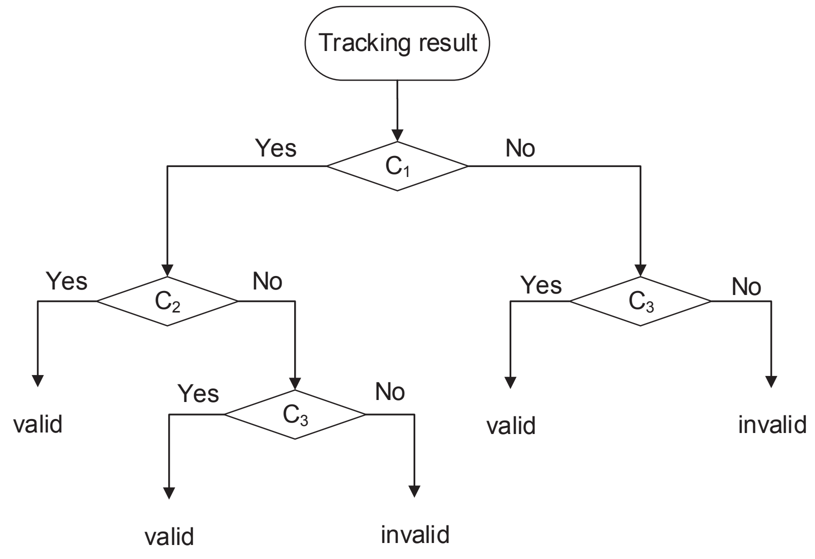


Figure D.2: Flowchart of consistency checking module