Alternative search techniques for face detection using location estimation and binary features

THÈSE N^o 5325 (2012)

PRÉSENTÉE LE
Á LA FACULTÉ SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
LABORATOIRE DE L'IDIAP
PROGRAMME DOCTORAL EN GÉNIE ÉLECTRIQUE
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES
PAR
Venkatesh Bala Subburaman

acceptée sur proposition du jury :

Dr. Jean-Marc Vesin, président du jury
Prof. Hervé Bourlard, directeur de thèse
Dr. Sebastien Marcel, co-directeur de thèse
Prof. Abdenour Hadid, rapporteur
Dr.-Ing Hazim Kemal Ekenel, rapporteur
Prof. Jean-Philippe Thiran, rapporteur

Lausanne, EPFL
2012

Abstract

The sliding window approach is the most widely used technique to detect objects from an image. In the past few years, classifiers have been improved in many ways to increase the scanning speed. Apart from the classifier design (such as the cascade), the scanning speed also depends on a number of different factors (such as grid spacing, and scale at which the image is searched). Scanning grid spacing controls the number of subwindows being processed, thus controlling the speed of detection. When the scanning grid spacing is larger than the tolerance of the trained classifier it can suffer from low detections. In this thesis, we propose an alternative search technique, which can improve the detections when lesser number of subwindows are processed.

First, we present a technique to reduce the number of miss detections while increasing the grid spacing when using the sliding window approach for object detection. This is achieved by using a small patch to predict the location of an object within a local search area. To achieve speed, it is necessary that the time taken for location prediction is comparable or better than the time it takes in average for the object classifier to reject a subwindow. We use binary features and a decision tree as it proved to be efficient for our application. In the process we also propose a variation of an existing binary feature (Ferns) with similar performance, and requires only half the number of pixel access when compared to Fern feature. We analyze the effect of patch size on location estimation and also evaluate our approach on several face databases. Experimental evaluation shows better detection rate and speed with our proposed approach for larger grid spacing (lesser number of subwindows) when compared to standard scanning technique.

We also show that by using a simple interest point detector based on quantized gradient orientation, as the front-end to the proposed location estimation technique, we can achieve better performance even when fewer number of subwindows are processed. The interest points detected can be assumed as a non-regular grid compared to regular grid in the sliding window framework. A few image patches are sampled around an interest point for estimating the probable face location and further verified using a strong face classifier. Experiment results show that using an interest point detector can reduce the number of subwindows processed while maintaining a good detection rate.

Keywords: Face detection, location estimation, decision tree, binary features, interest point.

Résumé

Les approches à base de fenêtres glissantes sont celles qui sont les plus utilisées pour la détection d'objets dans les images. Ces dernières années, les classifieurs ont été constamment améliorés pour accélérer la vitesse de parcours de la fenêtre glissante et donc le temps total de recherche dans une image. En plus de la structure du classifieur (par exemple, une cascade), la vitesse de parcours de la fenêtre glissante dépend aussi d'autres facteurs comme le pas de la grille de parcours, à la fois dans l'espace image (position) et échelle (taille). Le pas de parcours contrôle le nombre de fenêtres sur lesquelles le classifieur est appliqué et ainsi influence fortement le temps total de détection. Augmenter le pas de la grille de parcours de la fenêtre glissante au delà de la tolérance du classifieur cause une augmentation des faux négatifs (détections manquées). Dans cette thèse, nous proposons une nouvelle méthode de parcours de l'image qui permet d'améliorer les taux de détection tout en traitant moins de fenêtres.

Dans un premier temps, nous présentons une méthode permettant de réduire le nombre de faux négatifs quand le pas de recherche de la fenêtre glissante est augmenté. Pour ce faire, une petite région de l'image (patch) est utilisée pour prédire la position de l'objet recherché dans un voisinage. Pour gagner en vitesse, il est indispensable que ce processus de prédiction soit rapide par rapport au temps que le classifieur met à rejeter une fenêtre (en moyenne). Pour cela, nous utilisons des caractéristiques à base de local binary patterns (LBP) et des arbres de décision qui ont été reconnus comme efficaces pour notre domaine d'application. Nous proposons aussi une variation de caractéristiques existantes (Ferns). Comparées aux Ferns, nos caractéristiques nécessitent deux fois moins d'accès aux pixels et produisent des taux de détection similaires. Nous évaluons l'impact de la taille du patch considéré (utilisé pour la prediction) et utilisons plusieurs bases de donnée de détection de visages. Nos résultats montrent que, par rapport aux méthodes standard à base de fenêtre glissante, notre approche a de meilleurs taux de détection pour des pas de parcours plus grand (moins de fenêtres à traiter).

Nous proposons aussi d'utiliser un detecteur de points d'intérêt (basé sur l'orientation quantifiée du gradient de l'image) pour pré-sélectionner les points en lesquels le prédicteur de position est appliqué. Cette approche permet d'obtenir de meilleurs taux de reconnaissance tout en appliquant le classifieur sur un plus petit nombre de fenêtres. Le detecteur de points d'intérêt peut être vu comme une grille non régulière, à la différence de la grille des méthodes classiques à base de fenêtre glissante. Quelques patchs sont choisi aléatoirement autour de chaque point d'intérêt pour estimer la position la plus probable de l'objet ; le classifieur est alors appliqué à cette position pour déterminer s'il y a effectivement un objet ou non. Nos expériences montrent que cette méthode à base de points d'intérêt réduit le nombre de fenêtres à traiter tout en conservant un bon taux de détection.

Mots-clés : détection des visages, estimation de la position, arbre de décision, les caractéristiques binaires, points d'intérêt.

Acknowledgements

I really enjoyed doing my PhD at Idiap. There are many people involved for making this happen. Firstly, I would like to thank my supervisor Sébastien for his guidance and support towards my thesis. He is very helpful and supportive at all times and I could discuss any issues with him. I would also like to thank, Hervé, my thesis director, for supporting me in different situations during my PhD, and also for his motivating talks on research.

My stay here at Martigny was made even more exciting by my friends: Deepu for his mouth-watering dishes and skiing; Dinesh, Jagan, Sriram, Sivaram, Joel, Harsha and Hari for badminton; Radu for table tennis and Anindya for his piano skills; Mathew for his helpful discussion; Lakshmi, Ramya, Murali, Gokul, Ashtosh, Majid, Stephanie, Elie, Nik, Serena, Anh-Thu, Tatiana, Patricia, Flavio, and Roger for giving me a good company; Remi for converting my abstract to French; Laurent, Siley Ba, Cosmin, Marco, Stefan and many others for really long discussion on research topics; Alex, Trinh-Minh-Tri, David, Paco, Chris, Roy, Joan, Valerie, Charles, Leonidas, and many more Idiap colleagues for the baby-foot.

I would like to also thank the admin at Idiap and EPFL for making all the process look very easy: Nadine, Sylvie, Corinne, Chantal, Christophe, and Ed. I should also mention the system guys: Bastein, Norbert, Philip, Frank and all others for providing a very smooth computing resources at all times. I would also like to thank Cyril for giving me an internship opportunity at MultiTel.

I thank my parents Subburaman and Somavalli, my sister Prabha, my parent-in-law Rajakumar and Sharmila, and my wife Abhilasha for the unconditional love, and support they have been providing. I would also like to thank my brother-in-law Umarengan, for discussing various technical topics.

Contents

1 Introduction **11**

1.1 Motivation 13

1.2 Objective 13

1.3 Contributions 13

1.4 Organization of the thesis 15

2 Related work **17**

2.1 General framework for face detection using the sliding window search 18

2.2 Face detection 18

2.2.1 Frontal face detection 18

2.2.2 Multi-view face detection 25

2.2.3 Evaluation of commercially available face detection systems 28

2.2.4 Discussion 29

2.3 Improving speed of search 29

2.3.1 Speed-ups for face detection 30

2.3.2 Speed-ups for other object detection 32

2.4 Summary 34

3 An alternative search technique **35**

3.1 Standard sliding window scanning technique 36

3.2 Proposed scanning technique 37

3.3 Patch classification 41

3.3.1 Classification and regression trees 42

3.4 Location estimation using a decision tree and binary features 44

3.5 Evaluation of location estimation on cropped face databases 47

3.5.1 Training and analysing the performance of a tree for location estimation 47

3.5.2 Location estimation for frontal faces 49

3.5.3 Location estimation for frontal faces with in-plane rotation 53

3.5.4 Location estimation for faces with out-of-plane rotation 55

3.5.5 Computational time of the tree 56

3.6 Summary 57

4 Face detection using location estimation **59**

4.1 Objective 60

4.2 Performance evaluation 60

4.2.1 Merging multiple detections 60

4.2.2 Measuring correct and false detections 63

4.2.3 Baseline Classifiers 63

4.3 Experiments on the CMU face databases 64

4.3.1 Description of the databases 64

4.3.2 Baseline frontal face classifier 65

4.3.3 Results on frontal face detection 66

4.3.4 Baseline frontal in-plane rotated face classifiers 73

4.3.5 Results on frontal in-plane rotated face detection 75

4.3.6 Baseline out-of-plane rotated face classifier 79

4.3.7 Results on out-of-plane rotated face detection 81

4.3.8 Discussion 84

4.4 Experiments on additional face databases 85

4.4.1 Cinema and Web frontal face databases 86

4.4.2 FDDB face database 90

4.4.3 CMU Multi-PIE database 93

4.5 Summary 97

5 On improving the proposed search technique 99
5.1 Improving the proposed search technique . . . 100
5.2 Interest point detection . . . 101
5.3 Experiments . . . 103
5.3.1 Experiment setup . . . 103
5.3.2 Evaluation on the CMU face databases . . . 107
5.4 Experiments on additional face databases . . . 111
5.5 Summary . . . 115

6 Conclusion and future work 117
6.1 Location estimation using a decision tree and binary features . . . 117
6.2 Application to face detection . . . 118
6.3 Possible directions for future work . . . 120

A Feature Extraction 123
A.1 Haar-like features . . . 123
A.2 Local binary patterns . . . 124
A.3 Ferns . . . 125

B Face detection using ferns 127
B.1 Binary feature selection using conditional mutual information . . . 127
B.2 Semi-Naive Bayesian classifier . . . 128
B.3 Experiments . . . 130
B.3.1 Database . . . 130
B.3.2 Training and testing . . . 131
B.3.3 Results for single stage classifier . . . 132
B.3.4 Cascade model architecture and results . . . 133

C Additional results on the FDDB face database 137

Curriculum Vitae 153

List of Figures

1.1 Illustration of different stages in face processing. 12

2.1 General framework for face detection using the sliding window approach. 19
2.2 Cascade architecture for speeding up the detection process. 23
2.3 Examples of multi-view faces. 25
2.4 Different detection structures for multi-view face detection. 26

3.1 Illustration of the standard scanning technique vs. our proposed scanning framework. 37
3.2 Estimated chance of hit with and without location estimation with respect to scanning grid spacing. 40
3.3 Estimated time vs. detection rate for the standard and proposed approach. 40
3.4 Example of face patches used for training a location estimator. 45
3.5 Illustration of a decision tree for location estimation. 46
3.6 Examples of some frontal face patches used for training a decision tree. 49
3.7 Performance analysis of location estimation for frontal faces without perturbation . . 51
3.8 Performance analysis of location estimation for frontal faces with perturbation 52
3.9 Examples of frontal in-plane rotated face patches for training a decision tree. 53
3.10 Performance analysis of location estimation for frontal faces with in-plane rotation. . 54
3.11 Examples of out-of-plane rotated face patches used for training a decision tree. 55
3.12 Performance analysis of location estimation for out-of-plane rotated faces. 56

4.1 Illustration of location estimation on a real image. 61
4.2 Illustration of intersection of two rectangles. 62

4.3 Merging multiple detections using ratio of area of intersection to joined areas. 62
4.4 Baseline ROC curves for different classifiers on the CMU frontal face database. 67
4.5 Baseline detection rate vs. grid spacing for different perturbation on the CMU frontal face database. 68
4.6 Detection rate vs. grid spacing with and without location estimation on the CMU frontal face database. 69
4.7 Detection rate vs. time with and without location estimation on the CMU frontal face database. 71
4.8 Some examples of frontal face detection on the CMU frontal face database. 72
4.9 Overview of our approach for frontal in-plane rotated face detection 73
4.10 Patch examples used for training rotation estimator. 73
4.11 Evaluation of in-plane rotation estimation. 74
4.12 Performance on the CMU rotated face database. 77
4.13 Some examples of frontal in-plane rotated face detection on the CMU rotated face database. 78
4.14 Overview of our approach for out-of-plane rotated face detection. 79
4.15 Patch examples used for training a profile estimator. 80
4.16 Evaluation of out-of-plane rotation estimation. 80
4.17 Performance on the CMU profile face database. 82
4.18 Some examples of out-of-plane rotated face detection on the CMU profile face database. 83
4.19 Performance on Cinema and Web databases. 87
4.20 Some examples of frontal face detection on the Cinema frontal face database. 88
4.21 Some examples of frontal face detection on the Web frontal face database. 89
4.22 ROC curves for the FDDB face database. 91
4.23 Performance on the FDDB face database. 91
4.24 Some examples of frontal face detection on the FDDB face database. 92
4.25 ROC curve on the Multi-PIE database 94
4.26 Performance on the Multi-PIE database. 95
4.27 Some examples of left profile detection on the Multi-PIE face database. 96
4.28 Some examples of frontal detection on the Multi-PIE face database. 96

4.29 Some examples of right profile detection on the Multi-PIE face database. 96

5.1 Illustration of our face detection approach for one interest point. 100
5.2 Examples of interest point detector on the images containing faces. 102
5.3 Surface plot of table 5.1. 104
5.4 Sampling of the image patches assuming that the interest points are located near the eye, nose and the mouth region. 106
5.5 Histogram of interest point for the CMU face databases. 106
5.6 Comparison of computational computation time vs. detection rate on the CMU face databases for different scanning approaches. 109
5.7 Some examples of face detection using interest points on the CMU face database. . . . 110
5.8 Comparison of computation time vs. detection rate on the Cinema and Web frontal face databases for different scanning approaches. 111
5.9 Some examples of face detection using interest points on the Cinema and Web face database. 112
5.10 Some examples of face detection using interest points on the Multi-PIE face database. 113
5.11 Comparison of computation time vs. detection rate on the Multi-PIE face databases for different scanning approaches. 114

A.1 Illustration of Haar-like features. 124
A.2 Computation of Haar-like features with integral images. 124
A.3 Illustration of LBP computation. 125
A.4 Different texture primitive detected by LBP. 126
A.5 Illustration of fern feature. 126

B.1 Face and non-face samples form training, validation and test dataset. 130
B.2 Illustration of fearn features overlapped on face image. 131
B.3 Classification error rate vs. number of fearn features. 132
B.4 ROC curve on the XM2VTS (normal) database. 133
B.5 ROC curve on the XM2VTS (darkened) database. 134
B.6 ROC curve on the database collected from web. 135

B.7 Some examples of the face detection results on the XM2VTS and BANCA databases . 135

C.1 ROC curve on the FDDB face database. 138
C.2 Some detection results on the FDDB face database. 139

List of Tables

2.1 Performance of commercially available face detection system on the CMU face test set with zeros false alarm. 29

3.1 Computation time to process a single subwindow for different depth of the tree. 57

4.1 Total number of subwindows processed for the CMU face databases for different grid spacing. 85

5.1 Total number of interest points and faces having 0 interest points lying within its bounding box on the CMU frontal face database for varying gradient magnitude threshold and region area threshold. 105
5.2 Evaluation of the detection rate, speed and the number of false detections using interest point on the CMU frontal face database. 107
5.3 Evaluation of the detection rate, speed and the number of false detections using interest point on the CMU rotated face database. 108
5.4 Evaluation of the detection rate, speed and the number of false detections using interest point on the CMU profile face database. 108

B.1 True and false positive rate for different test sets 133
B.2 Detection rate for various test dataset 134

Chapter 1

Introduction

Computer vision applications are now ubiquitously found in mobile phones, digital camera, cars, toys, hospitals, airports, and security areas, and this has been possible by the availability of fast processors and robust algorithms. One of the problem in computer vision is face detection. The goal of *face detection* is to detect the location and size of all faces in an image. This is an easy task for humans, indeed babies are able to recognize human faces very early. However, this task becomes challenging for a machine as the face image captured with a vision sensor, gets altered by pose variations (rotation out-of-plane), camera angle, illumination, facial expressions, and occlusions (glasses, sunglasses, hat).

There are many closely related problems of face detection. A general view of different face processing problems are shown in Figure 1.1. *Facial feature detection* aims at detecting the presence and location of features, such as eyes, nose, nostrils, eyebrow, mouth, lips, and ears, with the assumption that there is only one face in an image. *Face recognition* aims to (1) compare an input image (probe) against a database (gallery) and report a match, if any (*face identification*), and (2) verify the claim of the identity of an individual in an input image (*face authentication*). *Face tracking* methods continuously estimate the location and possibly the orientation of a face in an image sequence, while facial expression recognition concerns identifying the affective states (happy, sad, surprised) of humans. Clearly, face detection is the first step in any of the above automated system. The face detection problem is one of the oldest problems in computer vision, dating as early as 1972 [85]. Though there has been good detection performance [84; 107] in the past few years, it is still

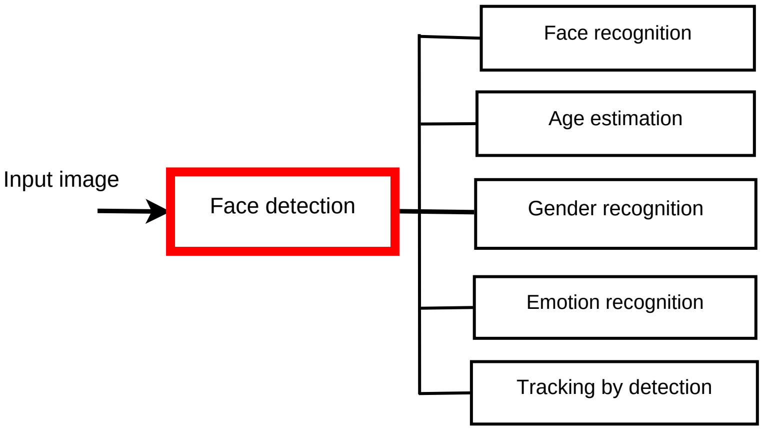


Figure 1.1. Illustration of different stages in face processing.

an interesting problem since recognizing patterns is not completely understood when compared to the performance of a human being.

In this thesis, we focus on the detection stage of the pipeline (Figure 1.1, the box shown in red). The usual and most widely used approach to detect objects from an image is the sliding window technique. A classifier with a fixed size subwindow is evaluated at every location and scale of the image, and an object is detected when the classifier response is above a preset threshold. The classifier has to evaluate millions of subwindows and has to be fast to achieve real-time performance. The cascade paradigm introduced by Viola and Jones [107] is one of the approaches to speed up the detection by rejecting the background as quickly as possible and spending more time on object like regions. Nevertheless, scanning with fine grid spacing is still computationally expensive. Cascade of classifiers speed up the detection to a limit which is difficult to overcome. One possible way to further speed up the detection process is to decrease the number of subwindows being evaluated, for instance, by increasing the grid spacing during the scanning process. Unfortunately, as the grid spacing is increased, the number of detections also decreases rapidly. This thesis focuses on alternative search techniques to reduce the number of miss detections when fewer number of subwindows are analysed.

1.1 Motivation

The number of vision applications in different digital devices are increasing. Each of these applications require an efficient processing of image sequences using more and more complex pattern recognition algorithms. Speeding up any of these tasks gives way to integrate more vision algorithms in a device. There has been different approaches to speed up detection algorithms. In face detection task cascade of classifiers has been the popular choice. In [47], object detection has been speeded up using branch and bound algorithm within a specific detection framework. In [18], the authors have cut down time consumed for feature computation by approximating the features at different scales. Some have implemented the existing approaches in GPU's which makes use of parallel computing to speed up the detection task [76; 13]. We try to come up with an alternative search approach targeting face detection which may add value to some of the above mentioned approaches.

1.2 Objective

In this thesis, we propose to answer to the question : " Is there some alternative search strategy to detect faces from an image which can perform reasonably good even when fewer number of subwindows are analysed ?". We split this problem into different subtasks:

1. We first investigate alternative search strategy in the sliding window framework.
2. To test our alternative search technique we evaluate our approach on frontal and multi-view face databases.
3. We investigate alternative binary features similar to ferns [68] and local binary pattern (LBP) [66] which can be used for classification task.
4. We also investigate interest points for quick focusing for face detection task.

We focus only on detecting faces from an image, but our approach should be equally applicable to detect other objects from an image.

1.3 Contributions

The main contributions of this thesis are as follows:

1. We propose **alternative search techniques** to reduce the number of miss detections when fewer number of subwindows are evaluated by increasing the scanning grid spacing in the sliding window framework. Our main idea is to combine Generalized Hough transform [7] within the sliding window framework to achieve the above mentioned goal. A small image patch is used to predict the location of the face region and is further verified by a strong classifier (cascade of boosted classifier) [93; 94]. We provide some theoretical insight on standard sliding window approach and the proposed one with respect to detection rate vs. grid spacing and detection rate vs. speed.
2. **Location estimation using a decision tree and binary features**: A decision tree is used to learn the association between the location (offset) and the patch (face) appearance. We investigate the performance of location estimation for different patch sizes, depth of the tree and for two different binary features (ferns and μ-ferns [93]) on cropped face databases. Next, we evaluate our proposed alternative search technique on various face databases (frontal face, in-plane and out-plane rotated faces) and show improvement in detection rate for larger grid spacing. The design of fast location estimation using a decision tree and simple binary feature offers speed to our approach without degrading the performance drastically.
3. **Improving the proposed search technique using interest points**: We also investigate the use of interest points which can be used to reduce the search space for face detection. The detected interest points could be considered as a non-regular grid as opposed to the regular grid in the sliding window approach. A few image patches are sampled around an interest point and probable face location are estimated using location estimation and verified using a strong face classifier. Any interest point detector could be used as long as some points on the face are detected for different face sizes and quality of the image. We propose a simple interest point detector based on gradient orientation which can be calculated very quickly. Our approach of using interest point along with location estimation shows good performance on different face databases.
4. Moreover, this thesis is proposing additional other contributions:

   – We explore the use of fern features for frontal face detection. The features are selected using conditional mutual information and are combined in probabilistic way to classify a given pattern as a face or non-face.

- We propose a new feature called μ-ferns which performs comparable to ferns. The difference between the two is that μ-ferns compares a pixel value to the average value of the patch, while ferns compare two pixel values. μ-ferns requires to access only half the number of pixels when compared to ferns, but requires an integral image for computing the average value of the patch efficiently.

Related publications

1. Bala Subburaman Venkatesh and Sebastien Marcel, “ Fast Bounding Box Estimation based Face Detection”, In ECCV 2010 Workshop on Face Detection: Where we are, and what next?, 2010. (**Achieved best paper award sponsored by Microsoft Research India** [1])
2. Bala Subburaman Venkatesh and Sebastien Marcel, “An Alternative Scanning Strategy to Detect Faces”, In International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2010.
3. Bala Subburaman Venkatesh and Sebastien Marcel, “Face Detection using Ferns,” https://publications.idiap.ch/index.php/publications/show/1575, Tech. Rep., Idiap-Internal-Com-06-2008.
4. Bala Subburaman Venkatesh and Sebastien Marcel, “A Fast Search Technique for Face Detection using Location Estimation”, (Journal submission) Tech. Rep., Idiap-Internal-RR-171-2010.

1.4 Organization of the thesis

The structure of this thesis is as follows:

- Chapter 2 first reviews different approaches to face detection. Then we review some of the work related to search and speed-ups for detecting faces and other objects from an image.
- Chapter 3 describes our alternative search technique of using location estimation for face detection within the sliding window framework. This chapter also discusses on training and testing a decision tree for location estimation. The performance of location estimation for

---

[1] http://vis-www.cs.umass.edu/fdWorkshop/

different patch sizes on cropped face databases and computation time for different tree depths are reported.

- Chapter 4 reports the experiment results of applying the proposed search technique on various face databases (frontal faces, in-plane and out-plane rotated faces) and compares with the standard sliding window approach. Different aspects such as detection rate vs. grid spacing and time vs. detection rate are discussed to bring out the advantages of our proposed search technique.
- Chapter 5 describes improving the proposed search technique by using interest points as a non-regular grid that focuses more on face region. A simple interest point detector is described based on quantized gradient orientation. Experiments are performed on benchmark face databases and compared with the standard sliding window technique.
- Chapter 6 concludes the thesis with a brief summary of contributions made and discusses the possible future research directions.
- In the appendix, we describe some other contributions of the thesis. This includes the work on fern feature for face detection. We also report the performance of profile face classifier on FDDB face database.

Chapter 2

Related work

Face detection is a mature topic and many different approaches have been proposed during the last decade. We will first present the basic and popular framework for face detection which is the sliding window approach in the next section. We will then review some of the work on frontal face detection in Section 2.2.1. Since images usually contain faces which are not always frontal, the need for detecting multi-view faces has also been addressed in the literature. We will review briefly some of the techniques for multi-view face detection in Section 2.2.2. A recent report on evaluation of commercially available face detection and recognition systems such as Picasa (Google), PittPatt (currently bought by Google), and iPhoto (Apple) is available in [21]. We briefly present only the results of face detection systems in Section 2.2.3.

With the accuracy level of object detection increasing, a number of applications that uses detection of faces/objects as front end has been growing and the need to process images quickly is gaining importance for real-time applications. The processing of high-definition (HD) images and videos is also becoming a challenging task. To speed up the search, different approaches have been proposed for faces, human/pedestrian, and other general objects, from which we can get a different insight into the area of object detection. We will review some of speed up techniques for object detection in Section 2.3, and finally, summary is presented in Section 2.4.

2.1 General framework for face detection using the sliding window search

The usual and the most popular approach to detect faces/objects from an image is the sliding window technique [84; 107; 16; 90; 109]. The general framework for face detection using the sliding window approach is shown in Figure 2.1. In this approach a classifier is evaluated at every possible location and scale (subwindow) of the image. Features are extracted from the image patch (subwindow) to reduce the effect of noise and intra-class variations before giving as input to the classifier. The subwindow is labelled as face if the classifier score is above a specific threshold. Usually multiple detections occur near the target region and are merged to obtain the final bounding box.

2.2 Face detection

In this section, we will first review work on frontal face detection followed by multi-view face detection. Then, we describe the result of evaluating commercially available face detection systems on benchmark face databases.

2.2.1 Frontal face detection

There are two main survey papers on face detection, one by Yang et al. [114], and another by Low et al. [32]. Face detection approaches as categorized by Yang et al. [114] are mainly knowledge-based, feature invariant approaches, template matching, and appearance based methods, which are briefly reviewed next.

(a) Knowledge-based and feature invariant approaches

Knowledge based approaches uses human knowledge about faces to derive the relationship between facial features. Yang and Huang [113] used a hierarchical knowledge-based method to detect faces. The rules are based on contrast between the face region and the environment around the face. The rules are split in 3 levels (similar to coarse-to-fine or cascade of rules) to efficiently search for face region. The first level rules are simple to detect roughly the face region. The second level

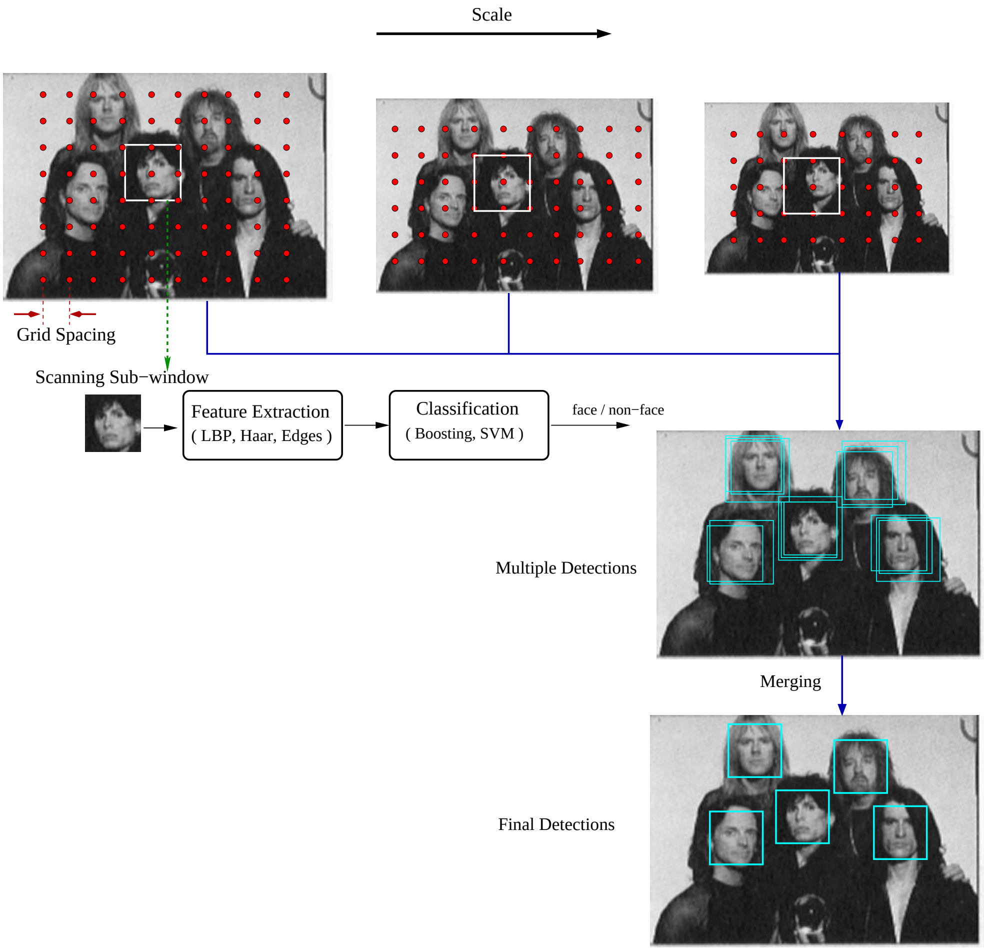


Figure 2.1. General framework for face detection using the sliding window approach. (a) The dots on the image represent the grid where the classifier is evaluated on a fixed sized subwindow. (b) Multiple detection in the neighbourhood of face in location and scale space. (c) Final detections after merging.

rules tries to detect the eyes, nose and the mouth based on vertical and horizontal sum projection of face region. The third set of rules operate on edges to further verify the presence of face region.

On the other hand, in feature invariant approaches, the algorithm tries to find structural features which exists even in the case of variation in pose, viewpoint or lighting conditions. The assumption is that there must exist features which are invariant to these variabilities. Usually edges or blobs representing the eyes, cheeks, mouth and the head region are extracted and their spatial configuration is verified for the face region [92; 12; 51]. In [51], the spatial distance between the facial parts are modelled using Gaussian distribution from an ensemble of training face images. Given a test image, candidate facial feature are detected by matching the filter response with the facial feature templates, and search for best constellation (eyes, nose, mouth) of face is formulated as a random graph matching problem in which the nodes correspond to the feature of face and the arcs represent the distance between different features. The constellations are ranked based on the probability density function that a constellation corresponds to a face, and a face is detected if there are more than 3 facial features found in a constellation. This type of model (pictorial structures for object recognition) was introduced by Fischler et al. [23] in 1973. The basic idea is to model the object as a collection of parts which are connected by spring like structure to allow for deformations to some extent.

Yow and Cipolla [117] found region of interest by first applying second derivative Gaussian filter to the image. Then the edges around these interest points are grouped to form regions. The grouping is based on edge strength similarity and orientation. The features from a region mainly edge strength, edge length and intensity variations are computed and matched against the stored feature vector for a valid facial feature. The facial features are further grouped and evaluated using Bayesian network to detect faces. The faces had to be larger than 60x60 for detection in their implementation. The main problem with feature-based algorithm is that the image features can be severely corrupted due to illumination, noise, and occlusion. Feature boundaries can be weakened for faces, while shadows can cause numerous strong edges which together render perceptual grouping algorithms useless.

Govindaraju et al. [105] approach, mainly operated on edges on which predefined rules are set to detect left, hair line, and right side curves of frontal face. Then pairs of feature are joined by edges if they could arise from the same face region. If the cost of a group of three feature curves with different labels are low then the group becomes a possible face hypothesis.

Another feature that has been used for face detection is the human skin color [34; 11; 17; 43; 100]. Different methods have been proposed to model the skin color in different color spaces (HSI, RGB, CIE, YCrCb). Mixture of Gaussian [35] and histogram methods [17] are often used to model the skin color. In [43], billions of skin pixels are used to model the skin color. In their experiments they find that histogram models to be superior to mixture models. One main advantage of color information is that it helps to reduce the search space by focusing on regions with likely skin color, but usually not sufficient to detect and track faces. It is usually combined with detecting oval shape of face and local feature analysis for verifying the presence of face region [34]. Unfortunately skin color cannot be used in gray scale images or when the faces are artificially colored (e.g., spectators having country flag painted on face), but can be powerful and greatly reduce the computation time in certain application scenarios.

Discussion

Though there has been different attempts based on feature invariant, knowledge and template based approaches, it still lacked the performance required for any real world application. In many of these approaches human expertise was used to design the hand crafted edge features and rules to detect faces from an image, which might not be sufficient for automatic detection by machine. Nevertheless, the edge features are powerful and are still used for object detection with various machine learning algorithms. For example edge-based grouping approaches for object detection are explored by Vittoro Ferrari et al. [22] and contour segment features along with Chamfer matching [9] have been used to detect horses, mugs, swans, humans and many other objects [75; 101; 28; 40; 60]. It can be said that, machine learning is one of the key components in improving the performance for various object detection algorithms. The advantage being that the knowledge about the object is learned implicitly by the machine learning algorithm during training time.

(b) Appearance-based approaches

Excellent results on face detection based on appearance-based approaches which relies on statistical analysis and machine learning lead research in this direction. In this approach a subimage of a fixed size is given to a classifier which takes a decision if it is a face or a non-face. Many different appearance-based algorithms have been proposed for face detection; Subspace methods [104], Neural Network [84], Support Vector Machines [67], Sparse Network of Winnows [81], Naive Bayes Classifier [88], and Information Theoretical approach [14]. Appearance based methods require the input image to be scanned at every location and scale. As a consequence the number of windows that needs to be tested easily reaches millions depending on the size of the image and the accuracy of the search. Therefore these methods need a classifier with a high true positive rate (detection rate), but also with an extremely low false positive rate (false alarm), typically of the order of 10^{-6}.

In the early 1990's, Sirovich and Kirby [46] developed a technique using PCA to efficiently represent human faces. Turk and Pentland [104] later proposed to perform Principal Component Analysis (PCA) on training face images and to use the eigenvectors, also called eigenfaces, as a face template. A candidate subwindow region is classified according to the distance computed in PCA subspace after projection. The distance can be interpreted as a measure of faceness. To better model the face space, Sung and Poggio [95] divided it into subclasses using mixture of multi-dimensional Gaussians and was decomposed in PCA subspace. A set of Mahalanobis-like and Euclidean distance were computed for a new image with respect to the face and non-face clusters and given to a trained Multi-Layer Perceptron (MLP) for face/non-face classification.

Rowley et al. [84] incorporated face knowledge in a retinally connected neural network and reported results on a difficult dataset. To improve the performance, multiple neural networks were trained and the output were combined with an arbitration strategy. Feraud et al. [78] suggested a different neural approach, based on a constrained generative model (CGM). Their CGM is an autoassociative fully connected MLP with three layers of weight. The idea behind this model is to force nonlinear PCA to be performed by modifying the projection of non-face examples to be close to the face examples. Classification is obtained by considering the reconstruction error of the CGM.

Osuna et al. [67] used Support Vector Machine (SVM) for face detection. A SVM with a polynomial kernel function is trained for face/non-face classification. One of the main advantage of SVM is that it can work with few training samples. Schneiderman and Kanade [88] describe a Naive Bayes classifier to estimate the joint probability of local appearance and position of face patterns (subregions of face) at multiple resolutions. At each scale, a face image is decomposed into smaller subregions to cope with small alignment errors during matching. These subregions are then projected to a lower dimensional space using PCA and quantized into a finite set of patterns, and the statistics of each projected subregions are estimated from the projected samples to encode local appearance. Their method decides that a face is present when the likelihood ratio is greater than the ratio of prior probabilities. They also extended this method with wavelet representation to detect profile faces and cars [89].

Roth et al. [81] used Sparse Network of Winnows (SNoW) learning architecture for face detection. The SNoW learning architecture is a sparse network of linear functions that utilizes the Winnow update rule [57]. Their face detector makes use of Boolean features that encode the positions and intensity value of pixels. Garcia and Delakis [27] proposed a convolution neural network for detecting semi-frontal human face in complex images. Their method automatically derives optimal convolution filters that act as feature extractors. This technique has also been used for other object detection tasks [96].

Boosting: All the above classification techniques provide good results for face detection but are computationally expensive as the input pattern has to go through all the calculations before making a decision for face and non-face. In 2001, Viola and Jones [107], proposed boosting Haar-like features with cascade architecture (ref. Figure 2.2) for face detection which achieves good performance in real-time. A recent survey on face detection mainly focusing on boosting approaches are presented in [119]. The main idea of boosting is to build a strong classifier by combining many weak classifiers [116]. A weak classifier only needs to perform better than chance. Adaboost selects a small set of weak classifiers from a larger set to build an efficient classifier. Though many weak classifiers are required to achieve high performance, the key idea is to build a cascade of classifier, where each classifier consists a set of weak classifiers. The number of weak classifiers in each stage is varied in complexity when moving forward in the cascade. The initial stages has smaller number of weak classifiers (i.e., number of features in each classifier) to reject majority of the background quickly and focus more on face like region in the later stages of the cascade, thus achieving higher detection speeds. With this breakthrough approach, many research papers focuses on boosting framework, cascade architecture, and features which are fast to compute and robust to illumination changes. Many different boosting strategies have been developed over last few years. In addition to various boosting algorithms, feature extraction that were simple to compute, robust to illumination changes, and capture the structure of the pattern efficiently was also explored.

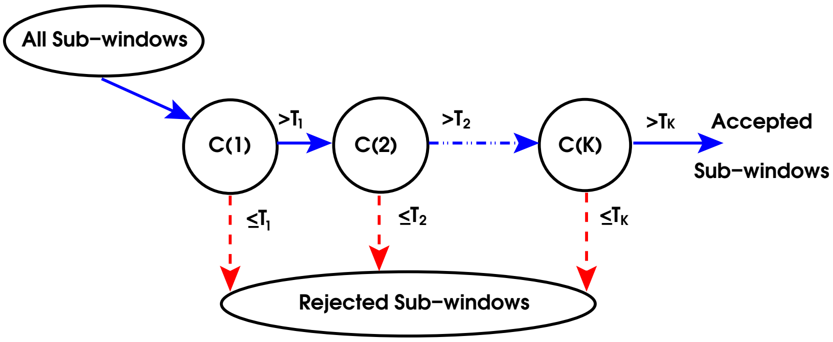


Figure 2.2. Cascade architecture for speeding up detection process. A series of classifiers $C(k), k = 1, ..., K$, are applied to every sub-window. A sub-window is rejected if it is lesser than the stage threshold $T(k)$, otherwise it is passed on to the next stage. The thresholds are selected such that large number of background sub-windows are rejected by first few classifiers.

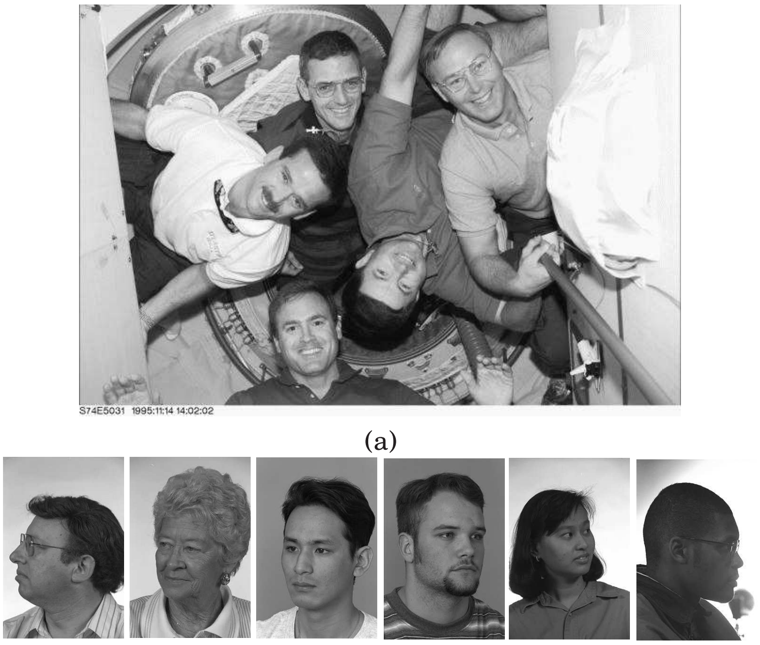
Jianxin Wu et al. [110] proposed a novel cascade learning algorithm based on forward feature selection which is two order of magnitude faster than Viola-Jones approach and yields classifier of equivalent quality. Huang et al. [36] proposed a nested cascade detector in which the confidence of the strong classifier from the previous layer is used as input along with other weak classifiers to the next layer. This reduces the number of layers and features used to reach the same detection and false positive rate. Stan Z. Li et al. [53] proposed Floatboost technique to backtrack and delete those weak classifiers that are non-effective or unfavourable in terms of error rate, which leads to a strong classifier consisting of fewer weak classifiers. In [59], a method to build a strong classifier with many weak classifiers without having go through a cascade architecture (number of stages, detection rate and false negatives for each stage, and number of weak classifiers for each stage) was proposed for training process which is termed as "softcascade". A cascade could be built without retraining by splitting the strong big classifier into several stages to achieve the required speed and accuracy. Similar to softcascade, Xiao et al. [111] proposed dynamic cascade which could be used to train on massive dataset. They also use multiple feature set (Haar-like , edge orientation histogram (EOH) [44], and Gabor features), and show that using different feature set can improve the classifier performance.

Another feature that has become popular mainly for face detection and recognition is the local binary pattern (LBP) [77] and its variants. Unlike Haar feature which gets affected due to illumination variation, the main advantage of LBP feature is that the feature response is not affected by monotonic transforms. Fröba et al. [25] used a variation of LBP called modified census transform (MCT) with boosting for face detection. In their approach the LBP feature was computed with comparison to mean gray value of the all pixels considered instead of center pixel alone. Lun Zhang et al. [122] introduced Multi-block LBP (MB-LBP) to further enhance the feature set for achieving better detection performance.

Recently in [103], extended set of local binary pattern was proposed for face detection; transitional LBP (tLBP) and direction coded LBP (dLBP) to encode additional pixel comparisons which did not exist in the original LBP pattern and also showed that a classifier built using the extended set of LBP pattern improved the performance of the detector by some amount. The percentage of different types of LBP used are also shown in their paper. An interesting observation is that modified MB-LBP (mMB-LBP; comparing to the mean value of all the pixels) dominates for face detection but for car detection tMB-LBP dominates.

2.2.2 Multi-view face detection

Faces usually do not appear frontal in the photos taken at different occasion. The face can have variation in pose and orientation, which adds additional challenges for learning and detection. Usually it is categorized into in-plane rotation (Figure 2.3(a)) and out-of-plane (Figure 2.3(b)). One could train a single classifier with all the pose and orientation variations, but this usually results in poor performance. Divide and conquer or coarse to fine strategy have been adopted by researchers to tackle this issue.



(a)

(b)

Figure 2.3. Examples of multi-view faces. a) in-plane rotation, b) out-of-plane rotation.

One of the simplest method is to train and search for each view independently. The parallel cascade shown in Figure 2.4(a) was proposed by Wu et al. [109]. They build a cascade detector for each view independently. The detectors are built only for few views (for example; frontal view with 60° and 90° in-plane rotation; left profile view with 60°, 90° and 120° in-plane rotation), and the rest of the detectors for other views are obtained by flipping and rotating the Haar features. To test if the new pattern is a face, all the detectors are run in parallel for first few stages. Then a decision is made to continue with only one classifier that has the highest output among others from the first few stages.

In [121], error correcting output codes are used to learn multi-class classifier. Each class is given a string of binary value and a SVM classifier is learnt for each of the binary bit. A test patch is passed through all the SVM classifier and a binary string is obtained. This binary string is then decoded to obtain the class label.

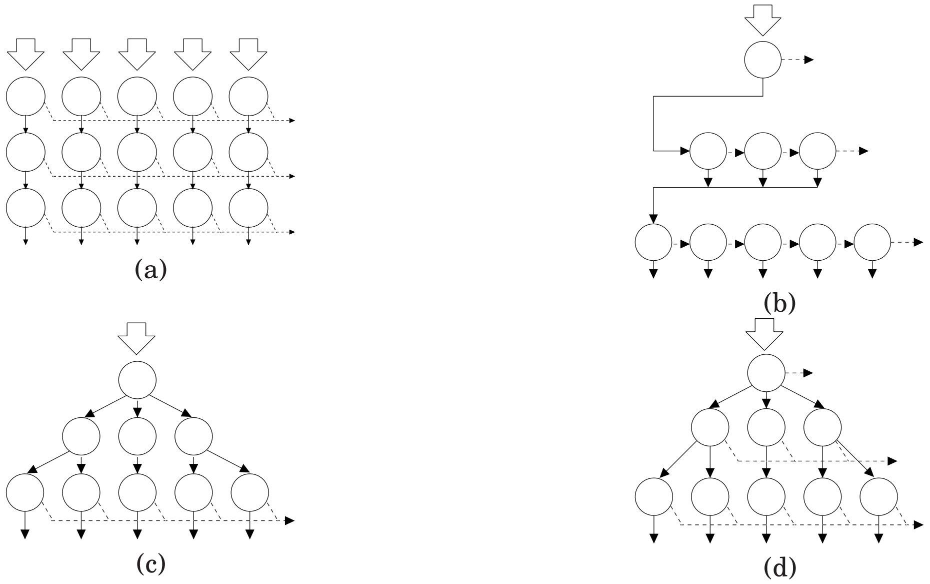

(a) (b) (c) (d)

Figure 2.4. Different detection structures for multi-view face detection. The solid lines represent the test input going to next stage, while the dashed lines represent that the input pattern is rejected. The circles represents a stage in a cascade. a) parallel cascade, b) detector pyramid c) detection tree, d) detection tree with early rejection. The circles represent strong classifier, solid lines are pass route, and dashed-lines are reject route.

Stan Z. Li et al. [54] proposed a detector pyramid architecture (Figure 2.4(b)) which adopts coarse-to-fine and simple to complex (top down in the pyramid) strategy to detect multi-view faces. Their architecture is designed to detect faces rotated out-of-plane $[-90°, +90°]$ with $+/-15°$ rotation in-plane. The pyramid has three levels. In the first level all the face poses are considered, while rejecting some non-faces. The second level is split into three pose categories $[-90°, -30°]$, $[-20°, +20°]$ and $[+30°, +90°]$. The last level is split into 7 pose categories. It is not clear why the test patch has to go through all the nodes in a level thus leaving room for further improvement in the detector design.

Viola and Jones [42] also proposed a fast multi-view face detector based on estimating the pose quickly by using a classification tree. Once the pose is estimated a view specific detector is used to verify the presence of face. The pose estimator architecture is shown in Figure 2.4(c). Rowley et al. [82] used a neural network classifier to estimate the rotation of face, and this information is used to normalize the input pattern and feed it to the upright face detector.

Huang et al. [37] presented vector boosting in which hard decision from a weak classifier is replaced with vector valued output. They allow the vector value output to be such that the nearby poses are considered for further exploration which is in contrast to [42] where a hard decision of pose is made. The features are shared in the initial stage of the architecture for different views, while at the end of the architecture the features selected are more view specific. In a similar work, Torralbe et al. [102] proposed a multi-task learning procedure, based on boosted decision stumps, that reduces the computations at run-time and complexity at training time, by finding common features that can be shared across classes (views). When the detectors for each class are trained jointly, the selected features are generic edge-like features and have good generalization (classifier performs well on unseen test patterns).

Most of the appearance-based approaches requires large amount of training data and manually labelling all the pose sub categories can be labour intensive. To tackle this issue, some researchers have approached this in an unsupervised method to automatically cluster the different views of face into sub categories. In [108], the authors proposed cluster boosted tree classifier, were a k-means clustering algorithm is used to split the samples into two sets when the discriminative power of a weak classifier becomes too weak. The classifiers are retrained once the samples are split into two parts. The features remain the same but only the classification function is recalculated, thus sharing features across different views. In a similar approach Babenko et al. [5] proposed multiple pose learning where the goal is to simultaneously split the data and learn a classifier for each view. They used LFW (Labled faces in the wild [38]) database to test their algorithm, though no performance results are provided on the benchmark face databases. Kim et al. [98], also proposed similar framework to cluster the samples during the training process. The initial sample labels are given using k-means clustering algorithm. After learning a weak classifier, the sample labels are re-evaluated by passing through the classifiers and labels are switched to the one that gives the maximum response. The idea is that if a pattern has similar feature response then it should belong to the same cluster.

Recently Zhang et al. [120], proposed a method to automatically cluster the multi-view face patches into different subcategories during the learning process itself. Their argument is that what is visually similar to human being might not be optimal for learning an overall detector. In similar approach to [98], the sample labels are switched during the learning process to improve the final classification performance, though the final resulting classifier will not be optimal in estimating the pose.

2.2.3 Evaluation of commercially available face detection systems

Here we present the detection performance of commercially available face detection mainly; Pitt-Patt (*http://www.pittpatt.com*, recently acquired by Google), Google Picasa (*http://picasa.google.com*), and Apple iPhoto (*http://www.apple.com/ilife/iphoto/*). The evaluation is done on CMU frontal face Test set A, B and C, CMU profile [89], and CMU rotated test set [82]. To evaluate the performance of face detection, images are imported from databases to the applications and the detected faces are checked whether they are faces and correctly framed. The performance of a face detection system is measured in terms of Detection Rate (DR), which is the proportion of the number of correct detection [1] to the number of faces present in the test set:

$$DR = \frac{\text{number of correct detections}}{\text{number of faces in test images}} \tag{2.1}$$

and the number of False Acceptance (nFA), which is the number of background regions detected. The results for CMU face test set are shown in Table 2.1. Note that the results are shown for zero false alarm. Pitt-Patt face detection system performs the best among the other two for the frontal, in-plane rotated and profile face databases. Picasa performs better for profile face detection compared to iPhoto, while iPhoto performs better for frontal and in-plane rotated face databases. The detection rate for profile and in-plane rotated faces are lower compared to the detection of frontal faces. Additionally it is clear from the results of the commercial system that the face detection is still not completely a solved problem.

---

[1] Each successful detection must contain the eyes and the mouth. The center of the detected bounding box must be around the center of the face.

	Pitt Patt	Picasa	iPhoto
CMU+MIT frontal face database			
CMU set A (169 faces)	140	140	130
CMU set B (155 faces)	123	69	122
CMU set C (183 faces)	178	170	151
Total (507 faces)	441 (87%)	379 (74.5%)	403 (79.5%)
CMU rotated face database			
CMU in-plane rotated (223 faces)	112 (50.2%)	44 (19.7%)	57 (25.5%)
CMU profile face database			
CMU profile (441 faces)	259 (58.7%)	260 (58.9%)	196 (44.4%)

Table 2.1. Performance of commercially available face detection system on the CMU face test set with zeros false alarm.

2.2.4 Discussion

Different algorithms and features have been proposed for face detection but, appearance based approaches perform the best and among different machine learning techniques, boosting with cascade architecture is usually adopted for this task. Among different feature sets, Haar and LBP are mostly used due to their computational efficiency.

Most of the appearance based approaches use sliding window technique to detect faces from an image. The number of subwindows that need to be processed can easily reach millions and can be computationally expensive. The idea of cascade of classifiers solves this issue to some extent by rejecting the background as quickly as possible and spending more resource on face-like region. Nevertheless, there has been attempts to improve the speed of search for object detection by many researchers. We will review some of the approaches in the next section.

2.3 Improving speed of search

The speed of detection of objects is of great importance for using it in many different applications that involves vision. The speed of search can be improved if we can use some property of the object that can be detected using some simple approach which is computationally less expensive and then applying the classifier within the interesting region instead of applying the classifier over all regions in the image.

2.3.1 Speed-ups for face detection

Prior to boosting and cascades of classifiers for face detection, Hoogenboom et al. [33] proposed a method to find interest point to speed up the detection process. They observed that due to illumination, the nose region was brighter than its neighbourhood, and therefore detect such region as the interest point. Based on the detected interest points, further analysis was carried out around this region to verify the presence of face.

Yali Amit et al. [3] proposed a two step procedure to detect face from an image. The first step is "focusing" in which small number of region-of-interest is found, and in the second step the selected regions are classified as face or background. In their work, focusing is based on searching for spatial arrangements of edge fragments (center of two eyes and mouth).

In [30], salient and stable feature such as the eye region is extracted based on the property that it looks darker than the rest of the face. An eye region pair is used to guide the search for faces. Their approach required that the face be at least of size 50x50 for successful detection. The advantage with such an approach is that it can be used to detect in-plane rotated faces without difficulty. Their approach would fail if one of the eye is occluded. The extraction of eye region based on its darkness property with respect to its surrounding can be compared to maximally stable extremal region (MSER) [19] which is used as interest point detector for object class recognition.

Rowley et al. [83] also proposed a two stage method to speed up the face detection process. The number of subwindows processed is related to the amount of translation invariance of the pattern recognition component. They trained a neural network to be tolerant to translation variation of the face pattern. To achieve this, a bigger subwindow than the face region was used to capture the translations of face pattern. This classifier was used to scan quickly (at a coarser grid spacing) the image to find likely region that contains a face. Once potential face regions are found, another neural network that has high performance is used to scan locally around to detect faces from an image.

In [87; 112], features computed in a subwindow are re-used in the overlapping window, and is termed as feature-centric approach. Computation of feature for each subwindow separately is termed as "window-centric". The idea of feature-centric approach thus saves computation time when compared to window-centric. Reusing features for neighbouring subwindows imposes some constrains on the feature type that can be used within a subwindow. The features extracted needed to be of the same type to be used in the neighbouring subwindow (i.e., the feature used at location i in one subwindow is same for location j for neighbouring subwindow). Though the feature computed is re-used, the classifier function for the feature extracted at different location can be different (i.e., different weight or look up table response based on its location). Said in another way, the features are computed beforehand and the classifier is evaluated using sliding window approach. To reduce the number of feature computation, Schniderman et al. [87] used feature functions that are computationally efficient in the initial stages of the cascade and used more complex feature functions in the later stages to be more discriminative.

Recently, Sznitman et al. [97] proposed a method to prune the search space quickly and reduce the number of strong classifier evaluation for final verification. Their technique follows a decision tree kind of approach termed as "Active Testing" where questions are asked in a sequential and adaptive way that are general and specific to face pose (location and scale) and feature space. The image is decomposed into a finite quad-tree and every leaf is associated with a pixel in the image. The non-terminal node corresponds to a unique subwindow in the image, representing a subset of face poses. The probability of face center to be within and not within a non-terminal node is learnt during the training process for all the queries. The queries are based on directional edge counting over a rectangle. During the search process each non-terminal node is updated to represent the probability of face center to be within the node. Instead of going through all the nodes in the next iteration, the most likely nodes based on the probability value are stored. Finally the node with highest probability is verified using boosted cascade classifier. To detect multiple faces, the edges from the current detections are removed and the whole process is repeated again. They report considerable gain in speed for larger image sizes when compared to sliding window approach, but for image size that are smaller than 112x74, they report that their approach is slower than sliding window approach due to computation overhead.

Butko et al. [10], proposed a method to improve the running time of object detector based on model of visual search in humans, which schedules eye fixations to maximize the long-term information acquired about the location of the target of interest. An image of any size is mapped to 21x21 grid cells, where the digital fixation occurs. For each fixation a digital fovea is simulated by having 4 image patches with each image patch covering a certain number of grid cells. Each image patch is rescaled to smallest size of grid cell. An object detector like Viola and Jones [107], is used at all location and scales for each of the image patches and the number of objects detected is used to guide the next placement of fixation on the grid. They report a speed up factor of 2 when compared to OpenCV[2] face detector. In their work at any time only one face could be fixated and it is not clear, how long it takes to detect multiple faces from an image.

2.3.2 Speed-ups for other object detection

Speeding up detection process is not only limited to face detection and it is of great interest for object detection that requires real-time computer vision applications. Pedestrian detection is one of them which is required in many visual surveillance systems and other applications.

In [123], a multi-resolution framework was proposed to detect pedestrians from an image. A separate classifier is trained for each object resolution, and during the detection phase, the higher resolutions are used only if it passes through the lower resolution classifiers. At higher resolution more number of orientation bins, smaller block sizes and higher sampling density of HOG feature were used to learn a classifier for better discriminative power. They also report that their approach leads to better recognition rate when compared to a single full resolution classifier, and can achieve detection speed of the lowest resolution classifier. The approach in [123] did not use any spatial constraint of object location which was considered in [70] to further improve the detection speed. In [70] the feature extraction parameters for each resolution is kept the same, thus avoiding extra computation when compared to [123] approach. Also where to focus next in each resolution is based on taking the local maxima response of the classifier around a neighbourhood. They report better speed of detection compared to sliding window approach. Felzenszwalb et al. [72], proposed a method on partial hypothesis pruning to speed-up deformable object detection in a star structured pictorial structure model. In [71], the authors presented a coarse to fine approach to speed-up deformable object detection, which is based on the observation that the bottleneck in part based object detection is in matching the parts rather than finding the optimal part configuration. The local maximum around a neighbourhood of lower resolution part classifier response guides the placement of higher resolution part. To further compensate for occlusion, blurring and other sources of noise, additional geometrical constrains between sibling of parts are enforced.

Computation of feature also adds to computational cost if it has to be evaluated at each scale.

---

[2] http://opencv.willowgarage.com/

In [18], the authors proposed a method to approximate the feature computed at one scale to nearby scales. This allows to decouple the sampling of the image pyramid from the sampling of detection scales, thus improving the speed of detection.

Zhou et al. [124], proposed shape regression machine to segment in real time an anatomic structure that manifests a deformable shape in a medical image. They split the problem into two stages. In the first stage parameters such as translation, scale and rotation are estimated using regression by sparsely scanning the image patches. The estimated object center is used as initialization for the second stage where a non-linear regression is used to predict the non-rigid shape from image appearance. Their approach achieves improved speed since only few image patches are scanned to estimate the object center or initialization point. Similarly, in [4], a multi-class random regression forest is used for efficient and automatic detection and localization of anatomical structures in a three-dimensional CT scan. Ozuysal et al. [69], proposed a method to localize a multi-view object (in their paper a car rotating out-of-plane is used for evaluation) using pose estimation. In the first step the aspect ratio and size of the object are estimated by first analysing a fixed sized subwindow. Then the viewing angle is predicted based on the hypothesis that the estimated bounding box contains the object. Finally a view specific classifier is used for verification.

Lampert et al. [47] proposed an elegant approach to localize object using branch and bound algorithm. The main element is the bounds on the sets of hypothesis, where a hypothesis represents a rectangle in the image. For example, if the features in a rectangle region in the image does not contain any features relevant to the object, the search can be terminated without further looking at smaller rectangles. The speed of convergence to global maxima depends on the quality of bounds. A tighter bound can lead to fast convergence, basically discarding the sets of rectangles quickly. The other advantage is the detected rectangle need not be of a fixed aspect ratio (usually in sliding window approach the aspect ratio is fixed to reduce the number of classifier evaluation). Neither the feature descriptor nor the classifier chosen matters as long as the decision function can be rewritten as a linear combination of individual contributions of each feature point. To detect multiple objects, the rectangle from the current detection is removed and the algorithm is run again to detect the next object. Following the branch and bound method, Lehmann et al. [48] proposed a feature-centric method for object detection, which reduces the memory requirement during detection phase.

Alternatively, parallel processing is gaining popularity to solve computer vision tasks. Graphics processing unit (GPU) have highly parallel architecture for fast operation on massive incoming data as in graphics rendering applications. This parallel architecture has been used by computer vision community to speed-up feature extraction, learning and detection of objects considerably. In [76; 13], object detection have been carried out in real-time by using GPU, which could achieve more than 60x speed-ups compared to standard sequential implementation.

2.4 Summary

We can see that there have been different attempts to speed-up the object detection task. The speed-ups are very clearly visible in some approaches when compared to others due to difference in the varying feature computation cost, scanning approach, type and complexity of the classifier used to detect objects from an image. The main idea behind speeding up detection process is to reduce the number of classifier evaluation. This requires efficient focusing to more object like appearance. In sliding window approach cascade of classifiers solves this to some extent. Simple or small number of features are used in the first few stages and in the later stages complex or more number of features are used to discriminate the object from the background. To further speed-up, feature-centric approaches have been proposed that avoid feature re-computation for neighbourhood subwindows and have shown improvement in detection speed.

Another approach for efficient searching is based on branch and bound technique and requires a bound on a set of hypothesis. Hough voting based approach [7] to object detection has been considered more natural or biologically possible [1; 2] when compared to sliding window technique. The key element is a feature votes for object center, which is similar to feature-centric approach. In [1], a connection between sliding window and Hough-based object detection is made explicitly, and show that branch and bound approach could also be applied to feature-centric view which is also memory friendly during detection phase. Detecting region of interest by simple computation is also another approach to quickly focus on object like region for further processing and can be seen as an alternative to sliding window approach. In short, some way or other all the techniques focuses on rejecting a set of hypothesis as quickly as possible. On the other hand parallel processing poses serious challenges for coming up with an alternative search techniques. But, we believe that any improvement in search technique will also be further benefited by parallel processing.

Chapter 3

An alternative search technique using location estimation and binary features

In this chapter we will focus on sliding window approach which is one of the most popular technique for face detection. A classifier is evaluated at every location and scale, and a face is detected when the classifier's response is above a preset threshold. Many systems need face processing tasks (detection, tracking, recognition), and needing them to run in real-time without loosing much of individual performance has become a challenging task. The cascade paradigm introduced by Viola and Jones [107] is one of the approaches to speed up the detection by rejecting the background quickly and spending more time on object like regions. Nevertheless, scanning with fine grid spacing is still computationally expensive.

Cascade of classifiers speed up the detection to a limit which might be difficult to overcome. One possible way to further speed up the detection process is to decrease the number of subwindows being evaluated by increasing the grid spacing during the scanning process. Unfortunately, as the grid spacing is increased the number of detections also decreases rapidly. Though it is known that the performance degrades when the grid spacing is made coarser, there is no literature which analyses this degradation.

Here we focus on achieving better detection rate for larger/coarser grid spacing by using a location estimator. We provide a theoretical insight on both the standard and the proposed scanning approach and is further validated with experiments on different face databases. We propose a location estimator using simple binary features and exploit the inherent multi-class nature of a decision tree which is also fast during test time. In this work we do not intend to increase the performance of the main face classifier, but rather try to improve the detection rate for larger grid spacing. Though our work focuses on faces, we believe that our approach could be applied to the detection of other object categories.

This chapter is organized as follows: In the following section we analyse the standard sliding window technique with respect to classifier and grid spacing. In Section 3.2 we present our proposed method using location estimator which is followed by discussion. Section 3.3 reviews briefly on patch classification followed by a brief introduction to classification and regression trees. In Section 3.4 location estimation using a decision tree is presented followed by evaluation of location estimation on face images. Finally, summary is presented in Section 3.6.

3.1 Standard sliding window scanning technique

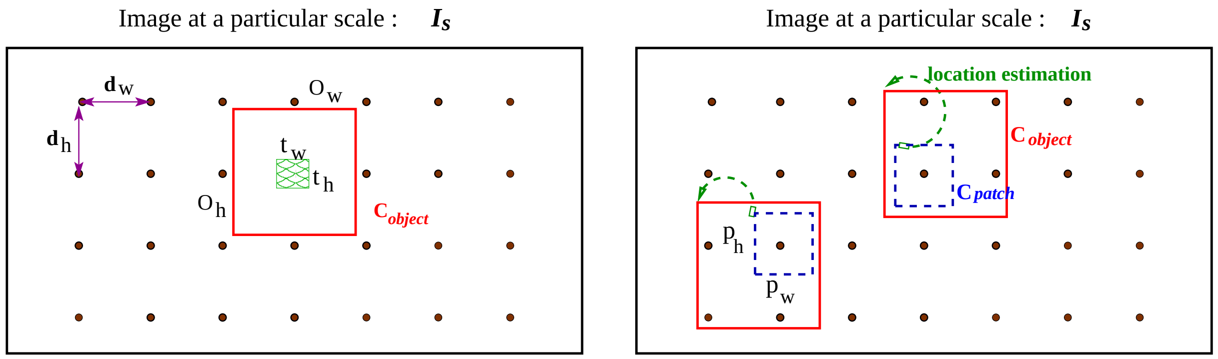
We follow pyramid based scanning approach as described in Rowley et al. [84]. The standard sliding window technique with regular grid scan for a single image scale I_s is shown in Figure 3.1(a), where a classifier C_{object} is placed on the scanning grid and checks if it is an object or not. The pseudo-code for scanning using sliding window is given in Algorithm 1. C_{object} is a strong classifier trained to detect objects of size (o_w, o_h). The classifier can fire multiple times near the object at a given scale, and the area within which it fires corresponds to the translation tolerance of the classifier. In Figure 3.1(a), (t_w, t_h) corresponds to the translation tolerance of classifier C_{object}. If a classifier has higher translation tolerance, the chance that the object is detected is high even when the image is scanned sparsely. We can start by formulating the chance of hit $\mathcal{H}_c$[1] of C_{object}, with respect to the scanning grid interval (d_w, d_h), and to the translation tolerance (t_w, t_h) of the classifier

---

[1] Note that it is not termed as probability as the value in (3.1) can be greater than 1.

C_{object},

$$\mathcal{H}_c \approx \frac{t_w t_h}{d_w d_h} \tag{3.1}$$



(a) (b)

Figure 3.1. Illustration of standard scanning technique vs our proposed scanning framework for a single scale I_s in image pyramid. The dots represent the scanning grid with interval (d_w, d_h), target object size (o_w, o_h), translation tolerance (t_w, t_h) of target object classifier C_{object}. (a) Standard scanning technique. (b) Our proposed scanning framework. Here (w_p, h_p) represents target patch size, and C_{patch} is the target patch classifier. The classifier C_{patch} predicts the location for C_{object} in our approach.

As an example, lets assume that the object present in the image is of the same size as the classifier is trained with, and if $t_w = t_h = 3$ and $d_w = d_h = 6$ then the chance of getting a hit $\mathcal{H}_c$ is 0.25, which is very low. For $\mathcal{H}_c$ greater than 1, means that the classifier has more chances to detect the object. As we decrease d_w and d_h (a finer search), $\mathcal{H}_c$ increases, while scanning speed decreases (slower). Our goal is to increase $\mathcal{H}_c$ without decreasing too much of the scanning speed (thus making it faster), which is described hereafter.

3.2 Proposed scanning technique

In this section we explain how our method increases the chance of hit. Figure 3.1(b), shows the proposed scanning framework for a given image scale I_s. The classifier C_{patch} is evaluated on a regular grid, while the main classifier C_{object} is placed on location predicted by C_{patch}. The pseudo-code for our approach is given in Algorithm 2. During training, C_{patch} learns the mapping between the appearance of face patch and its location (actually an offset) within the face region. At test time,

Algorithm 1 Face detection using standard sliding-windows.

```
input image : $I \in R^{W \times H}$; grid spacing : $d_w \geq 1$,$d_h \geq 1$;
scaling factor : $ds > 1$; starting scale : $S = 1$;
starting width and height : $W_s = W$, $H_s = H$;
trained object width and height : $O_w, O_h$;
classifier $C_{object}$;
set of detections: $D = \phi$
while $W_s > O_w$ and $H_s > O_h$ do
    scale the image: $I_s \leftarrow I \otimes S$
    for $x = 0$ to $x < W_s$ do
        for $y = 0$ to $y < H_s$ do
            $score = C_{object}\{x, y, I_s\}$
            if $score \geq \tau$ then
                $D \leftarrow D \cup \{x, y, S\}$
            end if
            $y \leftarrow y + dy$
        end for
        $x \leftarrow x + dx$
    end for
    $W_s \leftarrow \frac{W_s}{ds}$, $H_s \leftarrow \frac{H_s}{ds}$
    $S \leftarrow \frac{W_s}{W}$
end while
```

the patch is passed through the C_{patch} which predicts the location where the C_{object} is placed for final verification.Example of face image patches are shown in Figure 3.4. Assuming now that we have a classifier C_{patch} that predicts the patch location correctly within the translation tolerance (t_w, t_h) of the classifier C_{object}, with prediction rate D_p, then the chance of hit can be approximately given by:

$$\mathcal{H}_c \approx D_p \mathcal{H}_p \tag{3.2}$$

$$\mathcal{H}_p = \frac{(o_w - p_w + 1)(o_h - p_h + 1)}{d_w d_h} \tag{3.3}$$

where $\mathcal{H}_p$ is the chance of hit for the patch, (p_w, p_h) is the patch width and height, and (o_w, o_h) is the object width and height, with constraints $p_w < o_w$ and $p_h < o_h$ (see Figure 3.1(b)).

For example if, $p_w = p_h = 14$, $o_w = o_h = 19$, $d_w = d_h = 6$, and $D_p = 0.8$ (this value is taken from our experiment results), we get $\mathcal{H}_c$ = 0.8, which is 55% greater than standard scanning approach. Figure 3.2 shows the estimated chance of hit with and without location estimation with our formulation. We observe that as the patch size gets smaller and smaller we can achieve better detection rate for larger grid spacing.

Algorithm 2 Face detection using proposed approach.

```
input image : $I \in R^{W \times H}$; grid spacing : $dx \geq 1$,$dy \geq 1$;
scaling factor : $ds > 1$; starting scale : $S = 1$;
starting width and height : $W_s = W$, $H_s = H$;
trained object width and height : $O_w, O_h$;
classifier $C_{object}$; classifier $C_{patch}$
set of detections: $D = \phi$
while $W_s > O_w$ and $H_s > O_h$ do
    scale the image: $I_s \leftarrow I \otimes S$
    for $x = 0$ to $x < W_s$ do
        for $y = 0$ to $y < H_s$ do
            $\{x_{off}, y_{off}\} \leftarrow C_{patch}\{x, y, I_s\}$
            $score = C_{object}\{x - x_{off}, y - y_{off}, I_s\}$
            if $score \geq \tau$ then
                $D \leftarrow D \cup \{x - x_{off}, y - y_{off}, S\}$
            end if
            $y \leftarrow y + dy$
        end for
        $x \leftarrow x + dx$
    end for
    $W_s \leftarrow \frac{W_s}{ds}$, $H_s \leftarrow \frac{H_s}{ds}$
    $S \leftarrow \frac{W_s}{W}$
end while
```

We also look at detection rate vs time taken to process 1024x1024 image 200 times for different grid spacing with and without location estimation (see Figure 3.3). If we assume that it takes around 1μSec to process a subwindow using C_{object} and C_{patch} classifier to take $k \times C_{object}$ (k=1,0.5,0.33.0.25), we can obtain plots as shown in Figure 3.3. From the figure we can see that we can also gain in speed for a fixed detection rate when compared to without using location estimation.

Discussion The task boils down to identifying the image patch to infer the location from the face region. The smaller the patch size is, the more the spacing between the grid can be, for an increase in scanning speed. Unfortunately at the same time estimating the location becomes complex as individual patch will contain less and less information for distinguishing one from another. Matching or finding the similarity between two patches is one of the basic problem in pattern recognition, and it is challenging since the test patches can be a transformed and noisy version from the training set. Many different approaches have been proposed for identifying patches and we review some of them here next.

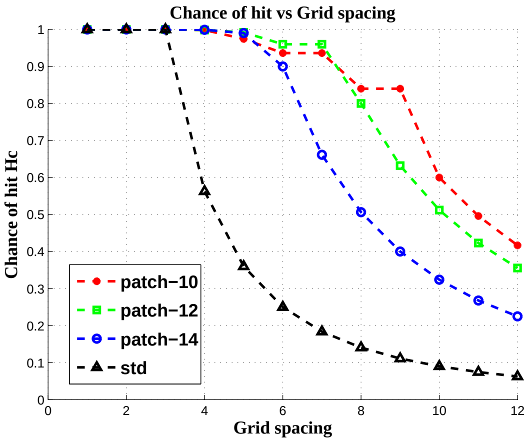
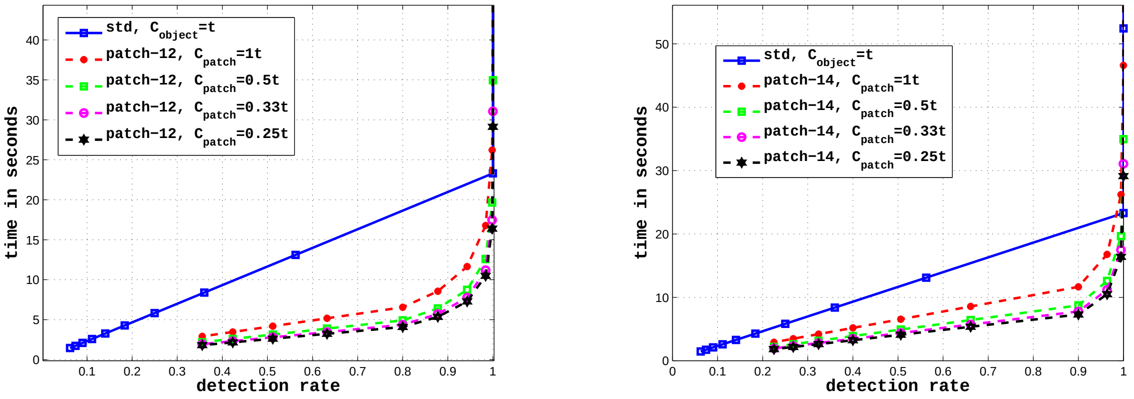


Figure 3.2. Estimated chance of hit $\mathcal{H}_c$ with and without location estimation with respect to scanning grid spacing. We show here how $\mathcal{H}_c$ varies for different patch sizes. In the figure's legend `-std` represent without location estimation, and `patch-10`,`patch-12`, and `patch-14` represents with location estimation for various patch sizes. The curves were generated using Eq (3.1) and (3.2), for $o_w = o_h = 19$, and approximate value for D_p (taken from experiment) were used for different patch sizes.



(a) (b)

Figure 3.3. Estimated time vs. detection rate for the standard and proposed approach.

3.3 Patch classification

The simplest method to match two image patches is using cross correlation or sum of squared difference, but to cope with transformed and noisy test patches, and to be computationally efficient different approaches have been proposed. SIFT (Scale Invariant Feature Transform) [58] is one of the most popular descriptors for matching two image patches extracted around an interest point detector. The descriptor consists of histogram of quantized oriented gradient. The descriptor is also rotation invariant by using the maximum gradient direction as the reference for each image patch. The image patches are matched by measuring the euclidean distance between the descriptors (Nearest neighbourhood search). Since computing Euclidean distance to all the stored descriptors can be time consuming, usage of kd trees and or reducing the dimension of the descriptor have been adopted. There are various extension of SIFT such as gradient location and orientation histogram (GLOH) [63] and PCA-SIFT [45].

To shift the computational burden at testing stage to the training stage, Lepetit et al. [106] proposed keypoint patch matching as a classification problem using randomized trees. To make the classification robust to pose and illumination changes, new views are synthesised from a small training set and serve as an input to tree-building algorithm. The test at each node are simple comparison of two pixel intensity values. To reduce the training time of the tree, instead of using classical approach to find the best binary test at every node using Entropy measures, the test at each node is randomized. They report that randomizing the test at a node decreases slightly the performance but greatly reduces the training time. The output of many randomized trees are averaged to improve the classification performance. In [68], the trees were replaced by ferns (the node test for each level of the tree is same), and a semi Naive-Bayesian classifier was used to classify the patches. They also show that ferns outperform trees for patch classification. In a similar framework of learning patches, Simon et al. [99] proposed histogram of quantized intensity for each pixel in the patch which is thresholded and stored in a compact binary representation for rapid computation of matching score using bitwise operation.

Patch-based object detection using generalized Hough transform has also gained popularity. One of the model proposed by Leibe et al. [49], Implicit Shape model (ISM), consists of class specific codebook of local appearance from the object category and spatial probability distribution of where the codebook entry may be found on the object. During recognition this information is used to perform a generalized Hough transform in a probabilistic framework. However as pointed out by Jurgen et al. [26], codebook-based Hough transform comes at a significant computational price, and the authors have suggested to directly learn a mapping using a random forest, between the appearance of an image patch and its Hough vote. More precisely, a probabilistic vote is obtained about the position of an object centroid. In a similar framework, J. Shutton et al. [91] extended Hough-forest approach to estimate the human pose from a single depth image. A huge realistic synthetic depth images of humans of many shapes and sizes in highly varied poses sampled from a large motion capture database are generated which is used to train a forest of deep randomized decision trees to learn the mapping between the depth image patches and body pose. Their approach achieves very high performance and speed for human pose estimation from depth images. In [20], a random regression forest is used to learn the mapping between the depth data and the face center in 3 dimension for different head pose. For estimating head pose at test time, the regression forest is applied on the surface patches to vote for face center. In [115], random trees are trained to learn a mapping between densely-sampled feature patches and their corresponding votes in a spatio-temporal-action Hough space, which is used for action recognition.

In general tree based classification and regression are an efficient way of mapping a complex input space to discrete or continuous output parameters. They are powerful, naturally handle multi-class problems and are fast, while remaining reasonably easy to train. The regression and classification accuracy is further improved by taking the average or majority votes from an ensemble of decision trees or forest of decision trees. Also, the test at each node of the tree are simple (pixel value comparison) and can be efficiently computed and have shown to perform well for patch identification. Motivated by the simple, fast and efficient performance of tree based approach for patch identification in different applications, we adopted the same for identifying face patches for location estimation, and in-plane and out-of-plane rotation estimation in our search algorithm.

3.3.1 Classification and regression trees

Tree-based classification and regression are supervised learning methods, where a set of training pairs (X, Y) are given. X can be a feature vector or a set of measurements, while Y can be respective class label for classification or continuous value for regression. The goal is to efficiently learn the mapping between the input and output given the training set, such that for a new sample the correct output is predicted.

Learning in a tree-based method proceeds from the root node in top-down fashion by recursively partitioning the samples into two subsets at each node by selecting a test, that leads to the greatest increase in node purity (how well the test separates the classes). In the case of classification this can be accomplished using an impurity function, which is a function of the proportions of learning sample belonging to the possible classes. These proportions will be denoted by $p_1, p_2, ..., p_{J-1}, p_J$. The impurity function should be such that it is maximized whenever a subset of X corresponding to a node in the tree contains an equal number of each of the possible classes (If there are the same number of c_1 cases as there are c_2 cases and c_3 cases and so on, then we are not able to sensibly associate that node with a particular class, and in fact, we are not able to sensibly favour any class over any other class, giving us that uncertainty is maximized). The impurity function should assume its minimum value for a node that is completely pure, having all cases from the learning sample corresponding to the node belonging to the same class. Two such functions that can serve as the impurity function are the Gini index of diversity [50]

$$g(p_1, p_2, ..., p_J) = 1 - \sum_{i=1}^{J} p_i^2 \tag{3.4}$$

and the entropy function,

$$h(p_1, p_2, ..., p_J) = -\sum_{i=1}^{J} p_i \ln p_i \tag{3.5}$$

For regression the split selected at each stage is the one that leads to the greatest reduction in the sum of the squared differences between the response values for the learning sample cases corresponding to a particular node and their sample mean, or the greatest reduction in the sum of the absolute differences between the response values for the learning sample cases corresponding to a particular node and their sample median. For example, using the squared differences criterion, one seeks the plane which divides a subset of X into the sets A and B for which

$$\sum_{i:x_i \in A} (y_i - \bar{y}_A)^2 + \sum_{i:x_i \in B} (y_i - \bar{y}_B)^2 \tag{3.6}$$

is minimized, where $\bar{y}_A$ is the sample mean of the response values for the cases in the learning samples corresponding to A, and $\bar{y}_B$ is the sample mean of the response values for cases in the learning samples corresponding to B. So split selection is based on making the observed response values collectively close to their corresponding predicted values.

If the nodes in the tree are continued to be created until no two distinct values of X for the cases in the learning sample belong to the same node, the tree may be over-fitting the learning sample and the classifier may not generalize well for future cases. On the other hand, if a tree has only a few terminal nodes, then it may be that it is not making enough use of information in the learning sample, and classification accuracy for future cases will suffer. Some of the parameters to stop growing a tree (further branching of a node) are minimum number of samples that arrive at a node, checking if the split no longer adds value to the output, and the maximum specified depth of the tree. In order to compare the prediction accuracy of various tree-structured classifiers, there needs to be a way to estimate a given tree's performance for future observations, which is sometimes referred to as the generalization error. An independent test data could be used to measure the performance of a tree for different parameter setting to select the best one. Cross-validation could also be used if the training data is not abundantly available.

3.4 Location estimation using a decision tree and binary features

In this section we describe how the C_{patch} classifier is built. We intend to use decision tree as it proved to be simple and efficient for our task. We will denote a tree with T. A decision tree is used to learn the association between the patches and its offset value. The term offset and location will mean the same in the rest of our thesis. Figure 3.4 shows some example of face image patches. We consider all overlapping patches within the face region. The key idea for our algorithm lies in estimating the location value with high performance (speed and accuracy).

We represent a set of patches by $\{\mathcal{P}_i = (\mathcal{I}_i, \boldsymbol{d}_i)\}$, where $\mathcal{I}_i$ is the appearance of the patch and $\boldsymbol{d}_i$ is the location parameter of the patch. The location parameter here is a 2D vector representing (x, y) shifts from the object center or from a fixed point in the object. The smaller the patch size for a fixed object size, more number of offset parameters needs to be learned and estimated. A tree is composed of nodes (leaf and non-leaf nodes). During training, a binary test is selected to minimize

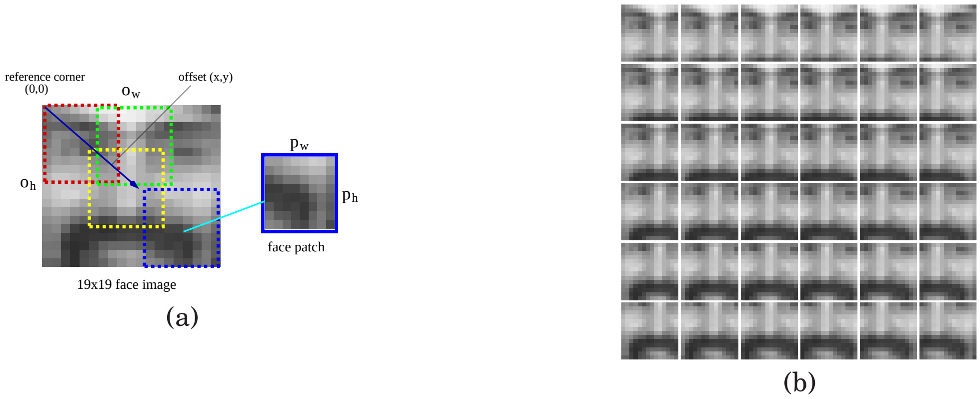


Figure 3.4. Example of face patches used for training a location estimator. (a) A face patch taken from within the face region. (b) All patches of size 14x14 extracted from the 19x19 face image shown in (a).

some error function in a non-leaf node, and in the leaf node location parameters are stored. The tree is trained in a supervised manner. Figure 3.5 shows a decision tree for location estimation. The binary test, tree construction, and data stored in leaf node are described as follows.

Binary test In a decision tree T, a test has to be performed at a node. We first consider a simple binary test introduced in [26; 68], which is given by:

$$t_f(\mathcal{I}) = \begin{cases} 1 & \text{if } \mathcal{I}(x, y) \leq \mathcal{I}(x', y') \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

where (x, y) and (x', y') are two locations in the patch $\mathcal{I}$. We also propose a new test termed as μfern [93] which is given by:

$$t_{\mu f}(\mathcal{I}) = \begin{cases} 1 & \text{if } \mathcal{I}(x, y) \leq avg(\mathcal{I}) \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

where $avg(\mathcal{I})$ is the average of the pixel values in the patch $\mathcal{I}$. Our test requires only half the number of pixel access compared to the previous test, but requires an integral image to quickly calculate the average value.

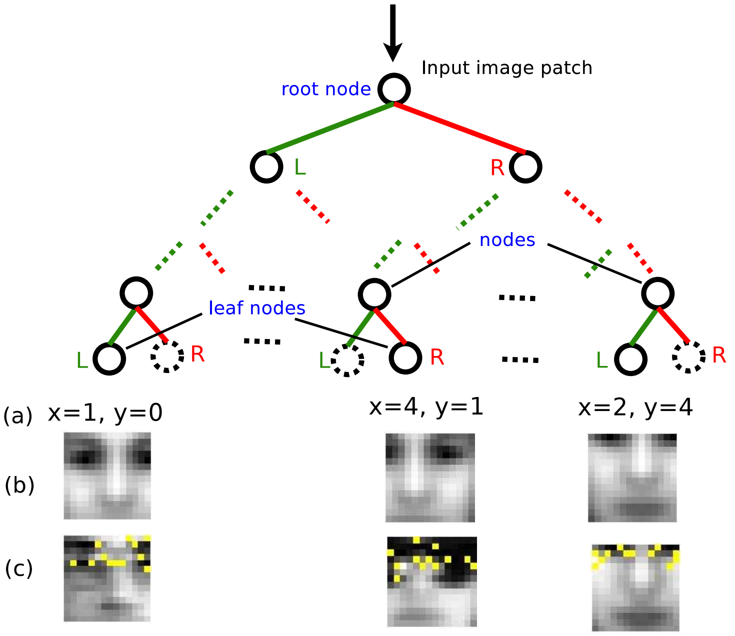


Figure 3.5. Illustration of a decision tree for location estimation. The patch size of 14x14 is shown here. An input image patch is passed through the root of the tree and reaches a leaf node. L and R represent left and right nodes of the tree. Row (a) provides the estimated offset value of the patches reaching a leaf node. Row (b) shows the average image of all the test patches having the offset value corresponding to the leaf node. Row (c) shows the average of the test images that arrives at a leaf node. The pixel locations that are evaluated ($t_{\mu f}$) at the nodes of a tree are also indicated for different leaf nodes in yellow dots.

Tree construction During training, each non-leaf node picks the binary test that splits the training samples in an optimal way. We use the offset uncertainty as in [26] which is defined as:

$$U(A) = \sum_{i \in A} (\boldsymbol{d}_i - \boldsymbol{d}_A)^2 \tag{3.9}$$

where d_A is the mean offset vector over all object patches in the set $A = \{\mathcal{P}_i = (\mathcal{I}_i, \boldsymbol{d}_i)\}$. A binary test $t^\star$ is chosen to minimize the following expression:

$$t^\star = \arg \min_{t=1,\ldots,T} (U(A_L) + U(A_R)) \tag{3.10}$$

where T is the number of possible binary tests, and A_L and A_R are the subset of training samples reaching the left node and the right node respectively under the binary test. Each leaf node l in the constructed tree stores a single offset vector (x_l, y_l). If A_l is the subset of training examples that arrives at the leaf node l, then (x_l, y_l) is given by:

$$x_l = \frac{1}{|A_l|} \sum_{k \in A_l} x_k \qquad y_l = \frac{1}{|A_l|} \sum_{k \in A_l} y_k \tag{3.11}$$

Similar to [26], we use two stopping criteria for the construction of the tree: the maximum depth of the tree and the minimum number of samples at a node. At runtime, a patch is given to the tree and the estimated offset value at the leaf node is used to place the main object classifier for subsequent detection. For illustration, we show in Figure 3.5 the pixel locations (x, y) associated to the tests $t_{\mu f}$ at each node in the tree from the root to different leafs. In these examples, we interestingly observe that the tree learns the shifts near the eye location.

3.5 Evaluation of location estimation on cropped face databases

We first briefly explain the training and measures used for evaluating location estimation. Then we show the performance of location estimation on cropped dataset for frontal, in-plane and out-of-plane rotated faces and discuss about computation time for different depth of the tree.

3.5.1 Training and analysing the performance of a tree for location estimation

Training a tree for location estimation requires patches from the face image. In our work we consider all overlapping patches within the face region and for each image of size (o_w, o_h) and patch size (p_w, p_h) there are $(o_w - p_w + 1)(o_h - p_h + 1)$ number of patches. Each patch is associated with an offset value which is measured from the top left corner of the face image (Figure 3.4(a)). We can safely assume that the patches have accurate ground-truth offset value, since all the faces are cropped with respect to the eye location.

Given a set of training patches $\{\mathcal{P}_i = (\mathcal{I}_i, \boldsymbol{d}_i)\}$, learning a tree proceeds recursively as described in Section 3.4. The number of tests at each node varies for the two different features (binary tests) considered in our evaluation. The total number of possible binary tests for t_f is $\frac{(p_w \times p_h)(p_w \times p_h - 1)}{2}$, which is large. Hence a fixed number of tests are evaluated (200 in our case) at every node, and for each test the pixel pairs are picked randomly as done in [68]. Among the 200 random tests at each node, the best is selected based on (3.10). The total number of possible binary tests in the case of $t_{\mu f}$ is $p_w \times p_h$, as it compares a pixel value to the average value of the patch. Since the number of test evaluation is small, each node evaluates all the test and selects the best one based on (3.10). The growing of the tree stops when a minimum number of samples reach a node, or the sample reaches the maximum specified depth. The minimum number of samples is set to 10 in all our experiments.

At test time, a patch is passed through the tree, and the leaf node gives an offset estimate $\hat{\boldsymbol{d}}(\hat{x}, \hat{y})$. We first evaluate the mean square error (MSE) for different depth of the tree on cropped validation database (different from training). The mean square error is given by

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{\boldsymbol{d}}_i - \boldsymbol{d}_i)^2 \tag{3.12}$$

where N is the number of examples, and $\boldsymbol{d}$ is the ground-truth offset values.

We also analyze how close the estimated offset is to the true offset. We define offset estimation error as:

$$\lambda = (\hat{x} - x)^2 + (\hat{y} - y)^2 \tag{3.13}$$

where (x, y) is the true offset value of the patch. We define $g(\lambda)$ as the number of test patches that have estimation error λ

$$g(\lambda) = \sum_{i=1}^{N} \mathbf{1}_{\lambda}(\lambda_{\mathbf{i}}) \tag{3.14}$$

where $\mathbf{1}_{\lambda}$ is the indicator function. To inspect the distribution of $g(\lambda)$, we look at the cumulative distribution $c(\lambda) = \sum_{j=0}^{\lambda} g(j)$. The cumulative distributions gives us an idea on the percentage of patches falling within a given offset error.

3.5.2 Location estimation for frontal faces

Cropped frontal face dataset

The face database we used are obtained from BANCA [74], BIOID [41], Purdue [61], and XM2VTS [62]. The ground-truth eye locations for these databases are publicly available. The faces are scaled and cropped with respect to eye location as described in [79]. The cropped face size can be changed by varying the distance between the eyes. In our experiments we used three different face image sizes (19x19, 19x27, and 24x24), to analyze different patch sizes for location estimation. The cropped faces are mirrored vertically to artificially increase the number of examples. A total of 45,000 cropped face images were obtained for each face image size. A subset of 15,000 face images are used for training, 15,000 for validation and the rest for testing. We create two different cropped dataset where one is called "normal", which does not have any perturbation, and another "rotscale" where the face images are perturbed with small amount of rotation (+/- 7 degrees from the center of cropped face image) and scaling (by changing the distance between the eye by +/-1 pixel). Figure 3.6, shows some examples of image patches of frontal view faces used for training the tree.

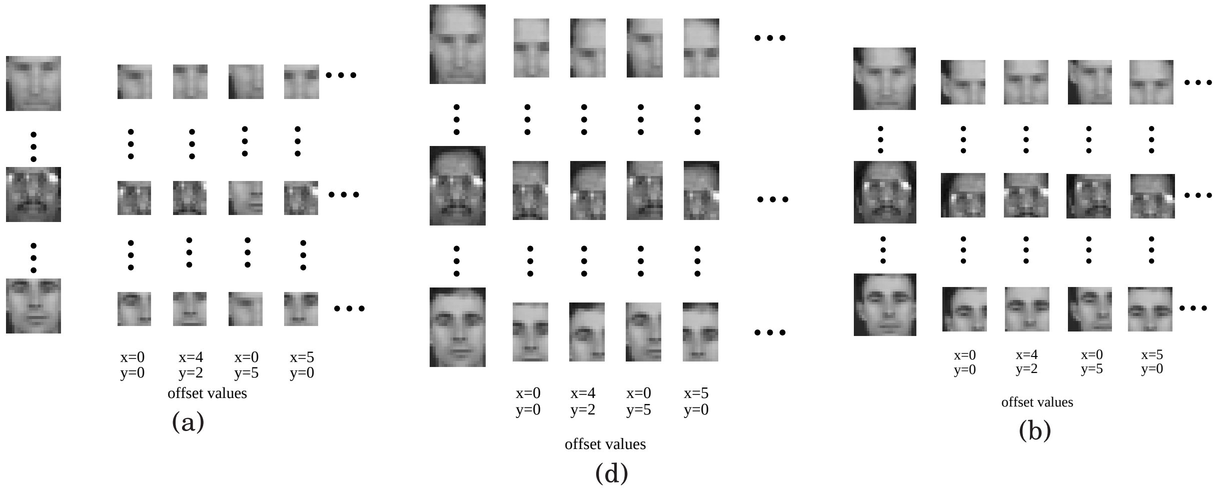
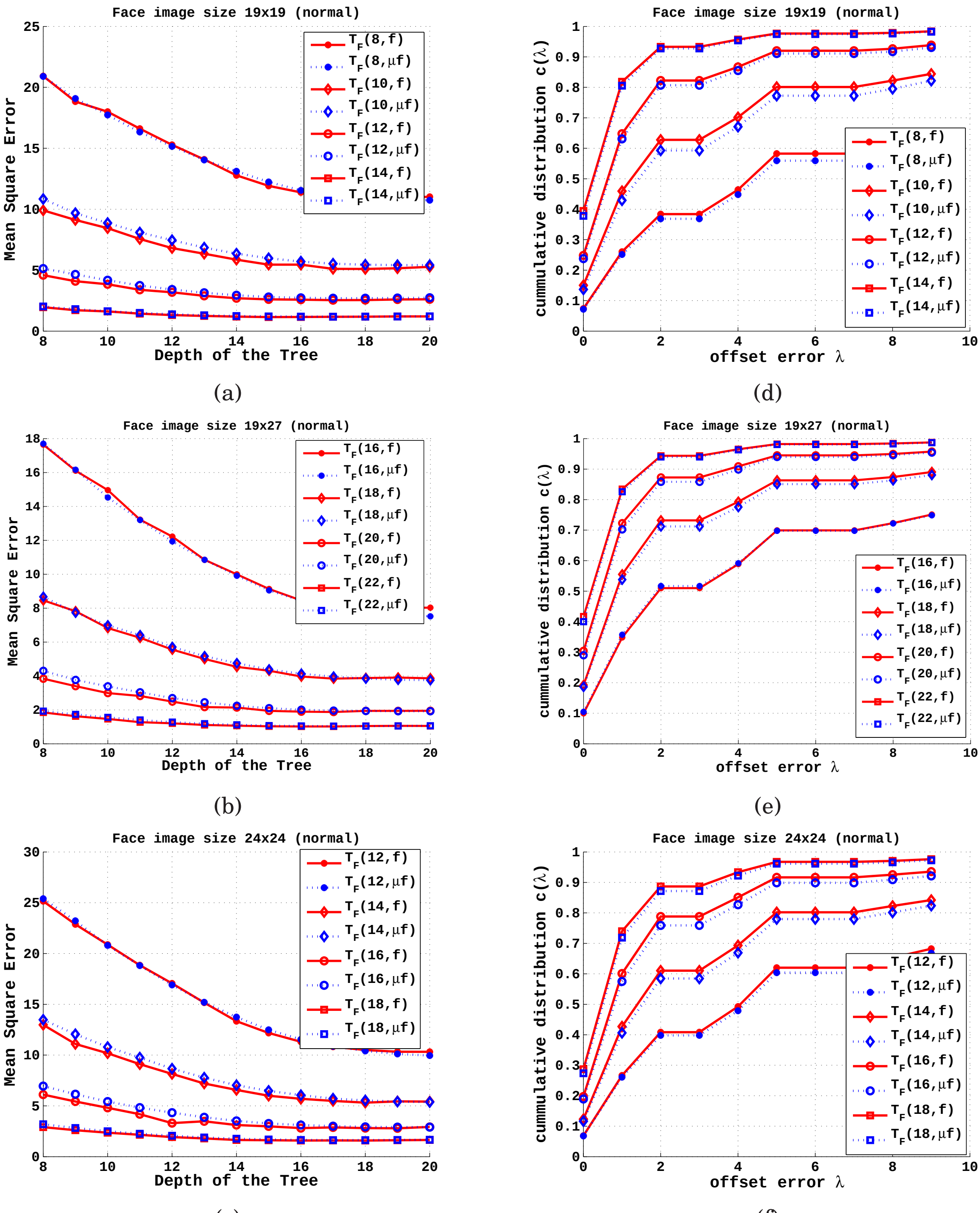


Figure 3.6. Examples of some frontal face patches used for training a decision tree. (a) 19x19 face image, (b) 19x27 face image, and (c) 24x24 face image.

Performance

We train trees with different depths (8 to 20) and for different patch sizes (for 19x19 face image: {8x8, 10x10, 12x12 and 14x14}, for 19x27 face image : {8x16, 10x18, 12x20, 14x22}, and for 24x24 face image :{12x12, 14x14, 16x16, 18x18}). A trained decision tree for location estimation will be represented by $T_F(p, b)$, where p represents the patch size (only height of the patch will be used), and b represents the binary test (μf or f). Figures 3.7(a,b,c) and 3.8(a,b,c) shows the MSE with respect to depth for different patch and face image sizes. From these figures we can see that for larger patch size it is sufficient to use smaller tree depth. For instance a patch size of 14x14 for a face image 19x19, a patch size of 14x22 for a face image 19x27, or a patch size 18x18 for a face image 24x24 have a similar MSE irrespective of the depth. In other words increasing the depth does not decrease the MSE. Hence retaining small depth size for larger patch size is sufficient. This is certainly because larger patch size have fewer offset values to be estimated. On the opposite smaller patch size have many offset values which requires larger depth size for better estimation. We also notice that irrespective of the face image size, the location estimation follows similar trend with respect to patch size and the number of offset values to be estimated. The MSE curves for the decision trees with the binary tests t_f and $t_{\mu f}$ are very similar. Note that as the depth of tree increases the computation time also increases. The appropriate depth of the tree can be selected based on required performance in speed and accuracy.

Figure 3.7(d,e,f) and 3.8(d,e,f) shows the cumulative distribution for different patches and face image sizes for a fixed depth of the tree. From the cumulative distribution we see that smaller patch sizes have less number of patches falling within a given offset error λ than for larger patch sizes. For larger patch sizes, it is possible to achieve around 80% to 95% of the test patches to fall within 2 pixel error (λ = 4). Also we see that there is not much difference in performance when trained with and without perturbation.



Face image size 19x19 (normal)

Face image size 19x27 (normal)

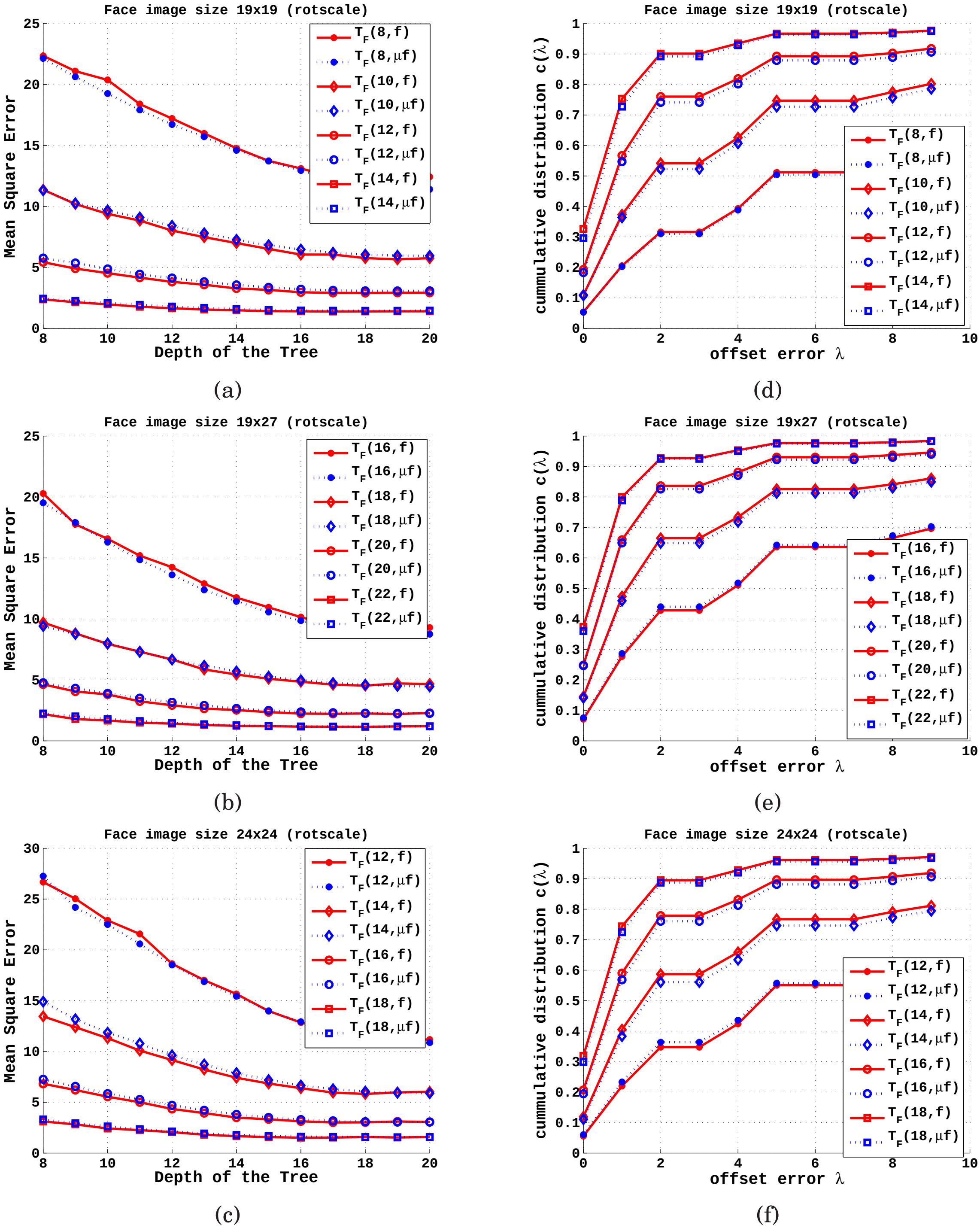
Face image size 24x24 (normal)

(a) (d)

(b) (e)

(c) (f)

Figure 3.7. Performance analysis of location estimation (without perturbation - "normal"). The face image size used is written on top of each figure. $T_F(p,b)$ represents a decision tree trained with patch size p and binary test b. (a,b,c) shows Mean Square Estimation Error vs Depth of Tree for different patch sizes. (d,e,f) shows the cumulative distribution $C(\lambda)$ for different patch sizes with respect to offset error λ. The tree depth is fixed to 15 for these plots.



(a) (d)

(b) (e)

(c) (f)

Figure 3.8. Performance analysis of location estimation (with perturbation - "rotscale"). The face image size used is written on top of each figure. $T_F(p,b)$ represents a decision tree trained with patch size p and binary test b. (a,b,c) shows Mean Square Estimation Error vs Depth of Tree for different patch sizes. (d,e,f) shows the cumulative distribution $C(\lambda)$ for different patch sizes with respect to offset error λ. The tree depth is fixed to 15 for these plots.

3.5.3 Location estimation for frontal faces with in-plane rotation

Cropped frontal in-plane rotated face dataset

We use the same cropped frontal face database as described in Section 3.5.2, and artificially rotate the image from -90° to $+90^\circ$ in steps of 10° to obtain training and testing data for in-plane rotated faces. For each rotation, 16,000 training examples are obtained. Considering all rotations from -90° to $+90^\circ$, we obtain 304,000 training examples. A square face image size is considered for this task, since it is easier to handle in-plane rotation. We have seen before that there is not much variation in performance between different face sizes and the results in Section 4.3.2 show that 24x24 face size performs better than 19x19. Therefore, 24x24 face size is selected for processing faces with in-plane rotation. Estimating the location for faces with in-plane rotation is also challenging due to the sharing of the same offset value for different patch appearances. Figure 3.9, shows some examples of image patches for frontal in-plane rotated faces used for training the tree.

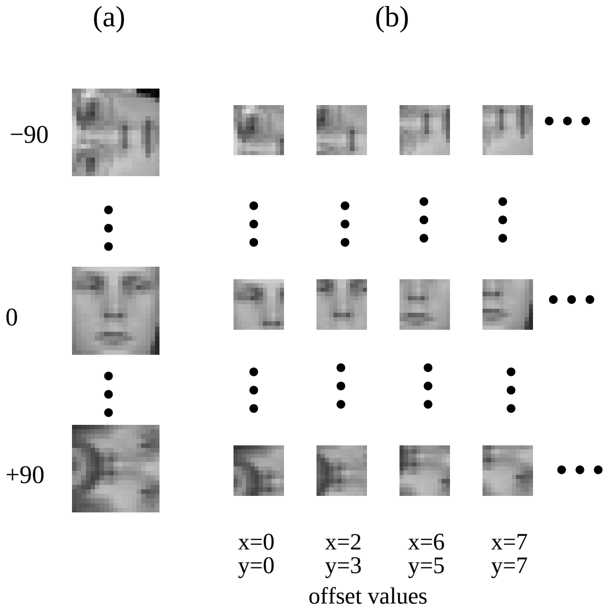
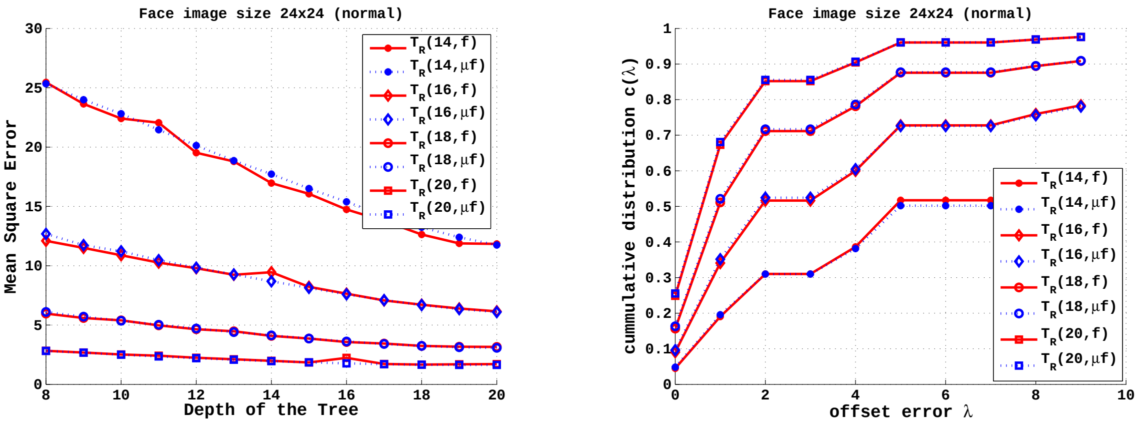


Figure 3.9. Examples of frontal in-plane rotated face patches for training a decision tree. Column (a) shows the original face image, and column (b) shows some patches cropped with respective offset values. The face image shown here is 24x24 and patch size is 18x18. The same offset value is shared by all different face orientation.



(a) (b)

Figure 3.10. Performance analysis of location estimation for frontal faces with in-plane rotation. The face image size used is written on top of each figure. $T_R(p, b)$ represents a decision tree trained with patch size p and binary test b. (a) shows Mean Square Estimation Error vs Depth of Tree for different patch sizes. (b) shows the cumulative distribution $C(\lambda)$ for different patch sizes with respect to offset error λ (depth of the tree is fixed to 17).

Performance

Similar to previous section, we train trees with different depths (8 to 20) and for different patch sizes (for 24x24 face image :{14x14, 16x16, 18x18, 20x20}). A trained decision tree for location estimation for frontal in-plane rotated faces will be represented by $T_R(p, b)$, where p is the patch size, and b represents the binary test used. Figures 3.10(a) shows the MSE with respect to depth for different patch sizes. The behaviour is similar to the tree trained for location estimation with just frontal face images. We also observe that for the same patch size training a tree with in-plane rotated images has higher error compared to training a tree with only frontal images. This is due to the sharing of different patch appearance for the same offset value (19 different appearance share the same offset value for this experiment).

Figure 3.10(b) shows the cumulative distribution for different patch sizes for a fixed depth of the tree. For larger patch sizes, it is possible to achieve around 80% to 90% of the test patches to fall within 2 pixel error (λ = 4). We also observe that there is around 10% less patches falling within the 2 pixel error when compared to Figure 3.7(f). We again see that the performance for two different tests t_f and $t_{\mu f}$ are very similar.

3.5.4 Location estimation for faces with out-of-plane rotation

Cropped out-of-plane rotated face database (profile faces)

We obtained profile face mainly from Feret database [73] which contains around 970 faces with 67° and 1300 faces with 90° out-of-plane rotation. We also collected from the web around 1250 profile (+/-90°) faces and annotated them with respect to eye and nose locations. The cropped data was artificially perturbed by small scaling (changing the distance between the visible eye and nose tip) and rotation ($+/-15^\circ$ from the center of cropped faces). The cropped faces are of size 24x24. Figure 3.11, shows some examples of image patches for out-of-plane rotated faces used for training the tree.

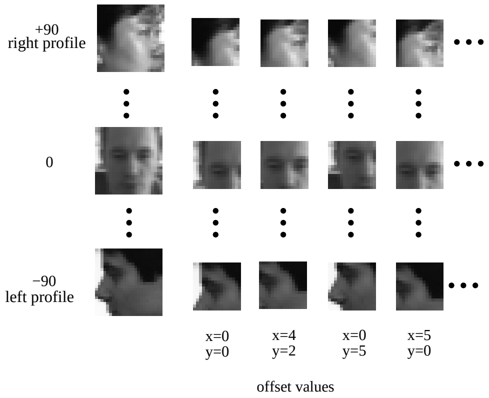
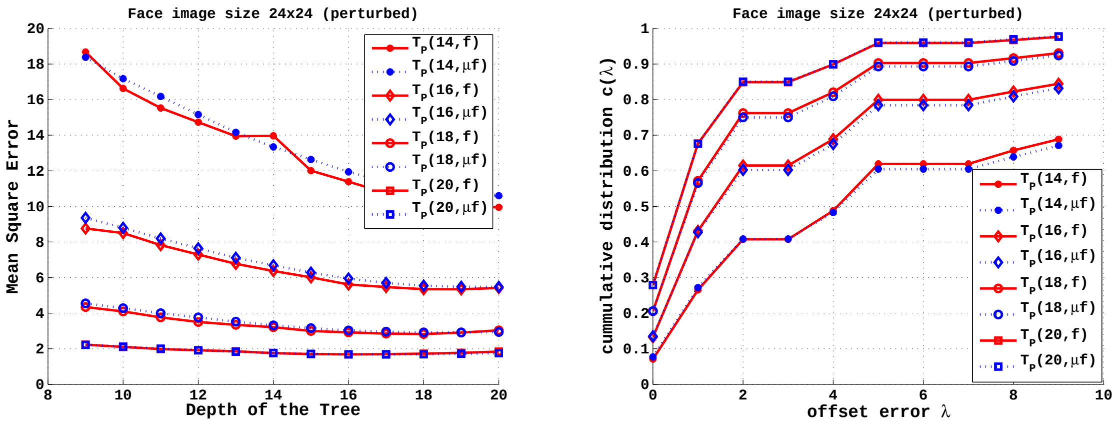


Figure 3.11. Examples of out-of-plane rotated face patches used for training a decision tree. Left most column shows the full face image (24x24), and on the right the cropped patches (16x16) with associated offset value at the bottom.

Performance

A trained decision tree for location estimation for out-of-plane rotated faces will be represented by $T_P(p, b)$, where p is the patch size, and b represents the binary test used. Figures 3.12(a) shows the MSE for different depth of the tree and patch sizes, and Figure 3.12(b) shows the cumulative distribution for different patch sizes (for 24x24 face image :{14x14, 16x16, 18x18, and 20x20}). The behaviours are similar to the previous two experiments. Here three different pattern appearances share the same offset value, but the variability of appearance for profile views (mainly due to different hair styles) are much larger than frontal in-plane rotation.



(a) (b)

Figure 3.12. Performance analysis of location estimation for out-of-plane rotated faces. The face image size used is written on top of each figure. $T_P(p,b)$ represents a decision tree trained with patch size p and binary test b.. (a) shows mean square estimation error vs depth of the tree for different patch sizes. (b) shows the cumulative distribution $C(\lambda)$ for different patch sizes with respect to offset error λ (the depth of the tree is fixed to 18 for this plot).

3.5.5 Computational time of the tree

We will now look at the computational time for the two different binary tests ($t_{\mu f}$ and t_f) for different depths of the tree. The computation time for a tree mainly depends on the depth of the tree which is the total number of binary tests performed. The binary test $t_{\mu f}$ requires integral image to compute the patch average. We will assume that the integral image is already computed for test $t_{\mu f}$ and hence computation of integral image time will not be considered. The only difference between the two test are the number of pixel access where t_f requires 2 pixel access for a binary test while $t_{\mu f}$ requires only one pixel access per binary test. To obtain the computational time we run the tree over CMU frontal face database several times for each depth of the tree and report the mean value in Table 3.1[2]. We can observe that a tree trained with test $t_{\mu f}$ is slightly faster than test t_f.

---

[2] The experiments were conducted on machine with Intel(R) Core(TM)2 CPU6700 @ 2.66GHz.

depth of tree	t_f	$t_{\mu f}$
11	$0.228\mu Sec$	$0.222\mu Sec$
12	$0.247\mu Sec$	$0.237\mu Sec$
13	$0.264\mu Sec$	$0.252\mu Sec$
14	$0.285\mu Sec$	$0.269\mu Sec$
15	$0.317\mu Sec$	$0.294\mu Sec$
16	$0.352\mu Sec$	$0.321\mu Sec$
17	$0.395\mu Sec$	$0.351\mu Sec$
18	$0.432\mu Sec$	$0.379\mu Sec$
19	$0.465\mu Sec$	$0.406\mu Sec$
20	$0.493\mu Sec$	$0.429\mu Sec$

Table 3.1. Computation time to process a single subwindow for different depth of tree and two different binary tests $t_{\mu f}$ and t_f.

3.6 Summary

In this chapter we proposed a method to improve the detection rate for larger grid spacing in the sliding window framework. The method is based on estimating the probable location of face by analysing a small image patch. Learning the mapping between the patch appearance and offset value is done using decision tree. For a fixed depth of the tree, as the number of offset parameters increases it becomes difficult to learn a tree with good location estimation. However, for a fixed grid spacing in the scanning process smaller patches have more chances to be within the face region compared to larger patches. The test at the nodes of the tree are kept simple and are fast to compute. We proposed a new binary test $t_{\mu f}$ which has similar performance to the binary test t_f irrespective of the face data used, and also the tree trained with test $t_{\mu f}$ is slightly faster than the test t_f.

Chapter 4

Face detection using location estimation

In this chapter, we evaluate our approach of using location estimation in a sliding window framework for detecting faces from standard face databases. Our goal here is to vary the number of subwindows processed by changing the grid spacing during the scanning process, and analyse the detection rate and scanning time. Analysing the detection rate by increasing the grid spacing gives us an idea on the detection performance when fewer number of subwindows are processed. Also, analysing the scanning time with respect to the detection rate will show if a better detection rate could be obtained for a fixed scanning time or obtain a better scanning time for a fixed detection rate. We will show that, our approach indeed achieves a better detection rate for a fixed scanning time or obtains a better scanning time for a fixed detection rate on different face databases.

This chapter is organized as follows. Section 4.1, illustrates how the location estimation modifies the search region for different grid spacings for an example image. In Section 4.2, the parameters used for scanning, for merging multiple detection and the measure to count correct and false detections are described. In Section 4.3, our approach is validated experimentally on the standard CMU face databases (frontal, rotated and profile) [90; 89]. Section 4.4 reports additional experiments on different face databases such as FDDB [39], Cinema and Web [27], and CMU Multi-PIE [29]. Finally, the summary of this chapter is presented in Section 4.5.

4.1 Objective

Our main idea of using location estimation as described in Section 3.2 is to increase the chance of placing the strong classifier on the target. In this section, we will visualize how the location estimation modifies the regular grid to increase the chance of finding a face in the image.

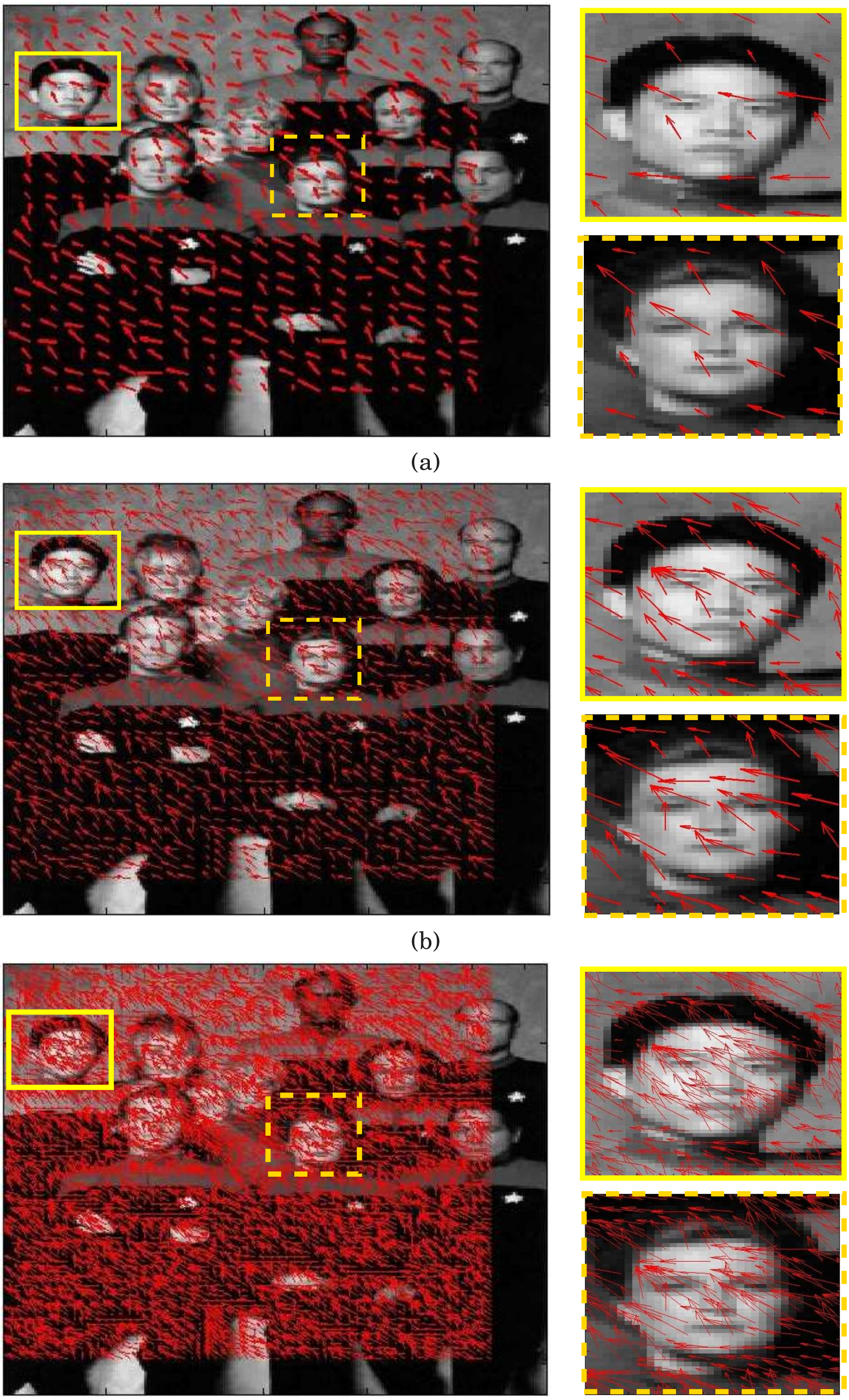
Figure 4.1 illustrates, how the location estimation guides the search to more face like regions for three different grid spacings (10 by 10, 6 by 6, and 3 by 3). To visualize the estimated location, arrows are overlaid on the gray image. The origin of the arrow indicates a point on the regular grid, while the arrow head indicates the new estimated location by C_{patch}. The arrows are pointing more or less in one direction because the top left corner of the face is used as the reference coordinate. For the location estimation, a decision tree $T_F(10, \mu f)$ for a face size of 19x19 is used for this illustration. From the zoomed image on the right of the Figure 4.1(a,b,c), we observe that the location estimation is able to predict the left top corner of the face region reasonably well for different grid spacings. For the grid spacing of 3 by 3, we can see that the location estimation is able to predict many times the top left corner of the face, and for a larger grid spacing (10 by 10) we still have one or two predictions on the top left corner of the face. Thus, using location estimation can increase the chances of a face being detected even when the grid spacing is made larger.

4.2 Performance evaluation

A standard pyramid based scanning approach (as in [84]) is used to detect faces at different scales. The scale factor to build the pyramid is set to 1.2. For location estimation, a decision tree with only μ-Ferns is considered in the rest of our experiments, since the performance was comparable to Ferns as seen from the previous chapter. In the next sections, merging of multiple detections, measuring correct and false detections and a brief description of baseline classifier used in our experiments are described.

4.2.1 Merging multiple detections

The merging of multiple detections is based on commonly used ratio of intersection/union of area [39]. Let us assume that a list of detections parametrized by a set of bounding boxes or rectangles


(a)

(b)

(c)

Figure 4.1. Illustration of location estimation on a real image. The base of the arrow is where C_{patch} is placed, and the arrow head is the offset estimate by C_{patch}. For this example the faces in the image are of size approximately 19x19. The images on the right show zoomed version of two faces. (a) shows location estimation for 10 by 10 grid spacing, (b) shows the location estimation for 6 by 6 grid spacing, and (c) shows location estimation on 3 by 3 grid spacing.

$\{\mathcal{D}_i = (x_i, y_i, w_i, h_i)\}$, where, x and y are the top left corner of the bounding box and w and h are its width and height respectively are given. Then the overlap ratio between the two rectangles is given by:

$$J(\mathcal{D}_p, \mathcal{D}_q) = \frac{\mathcal{D}_p \cap \mathcal{D}_q}{\mathcal{D}_p \cup \mathcal{D}_q} \tag{4.1}$$

where, $\mathcal{D}_p \cap \mathcal{D}_q$ and $\mathcal{D}_p \cup \mathcal{D}_q$ stands for the area of their intersection and reunion, respectively. The value of J ranges from 0 to 1 (1 for exact overlap and 0 for no overlap, see Figure 4.2).

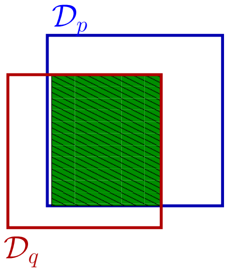


Figure 4.2. Illustration of intersection of two rectangles $\mathcal{D}_p$ and $\mathcal{D}_q$. The shaded region represents the intersection between the rectangles.

The merging of multiple detection is an iterative algorithm, where at each iteration some rectangles are merged, some are discarded and some are kept for next iteration. With this merging algorithm, three different overlapping regions (see Figure 4.3) could be controlled. The rectangles that have value J above a threshold value α_u, will be considered for merging. The rectangles that have value J below a threshold value α_u, and above a threshold value α_l will be discarded from the list, and the rectangles with value J below a threshold value α_l, will be kept for next iteration.

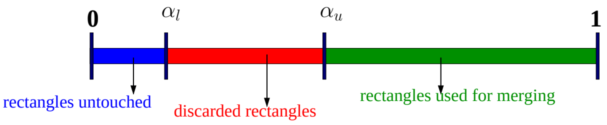


Figure 4.3. Merging multiple detections using ratio of area of intersection to joined areas. α_l and α_u are two threshold values that controls the grouping of rectangles into different region.

For merging multiple detections, when more than one view of the face is considered, the value J could be modified by multiplying by a function that reflects the closeness between two different views. Another parameter that can be used for accepting a merged detection is the minimum number of overlapping rectangles β. This is usually useful when the target is detected multiple times compared to the background, thus a higher value of β could be used to reduce the number of false alarms.

In all the experiments the value of α_u is fixed to 0.45, α_l is fixed to 0.05 for frontal and in-plane rotation while for out-of-plane rotated faces it is fixed to 0.1 (since the CMU profile face database contains faces occluded by other faces). The value β is set to 1, since for larger grid spacing the classifier might see the target only once.

4.2.2 Measuring correct and false detections

Once multiple detections are merged, the overlap ratio J between the rectangles could be used again to count the correct and false detection from the ground truth rectangles (bounding boxes). A face is said to be correctly detected if the value J is greater than the specified threshold α_T. The value for α_T is set to 0.35, since we are interested only by the detection task and not to realize a precise localization.

To analyse the performance of the system at a specific operating point (a determined α_T), the detection rate (DR) and the number of false alarms (nFA) are computed, which are given as:

$$DR = \frac{\text{number of correct detections}}{\text{number of true faces in the image}} \tag{4.2}$$

$$nFA = \text{total number of detections} - \text{number of correct detections} \tag{4.3}$$

To analyse the performance at different operating points, Receiver Operating Characteristic (ROC) curves could be plotted. ROC curve is a plot of the detection rate (true positive rate) with respect to the number of false positives (false alarms) at different operating points. Different operating points of the system are obtained by varying the final threshold of the baseline classifier.

4.2.3 Baseline Classifiers

A cascade of boosted multi-block LBP [122] features are used to train a strong view-specific classifier. A cascade architecture is chosen to reject the background as quickly as possible and focus on more face like regions, while LBP features are chosen for their robustness to monotonic illumination variations and are also fast to compute. The cascade architecture has 6 stages and the detection rate for each stage is set to 99.95%. The first stage has 5 weak classifiers, followed by 10, 17, 25, 40, and 60 weak classifier. The training of each stage is similar to the training described by Fröba et al. [25]. Negative patterns are updated after each stage by collecting the patterns that passes through the currently built classifier.

The baseline classifier, in the case of detecting only one view of the face consists of one strong view-specific classifier. In the case of detecting faces with more than one view (frontal in-plane rotated or out-of-plane rotated faces), a decision tree is trained to estimate the view of the face similar to the decision tree used for location estimation, and then a view-specific classifier is used for final verification (see Figure 4.9 and 4.14). Since, we are interested in analysing the effect of location estimation on the detection rate for different grid spacing, the baseline classifier for detecting more than one view will contain a view estimator and view-specific classifiers. This will be described in detail later in Sections 4.3.4 (for frontal in-plane rotated faces) and 4.3.6 (for out-of-plane rotated faces). All the experiments were conducted on machine with Intel(R) Core(TM)2 CPU6700 @ 2.66GHz.

4.3 Experiments on the CMU face databases

In this section, the performance (detection rate and speed) of scanning with and without location estimation for different grid spacing are compared on the CMU face databases [1]. First, a brief description of the face databases used are presented. Then, the baseline classifiers and results are presented for frontal, frontal in-plane rotated, and out-of-plane rotated face databases. Finally, we discuss on the computation time of the baseline classifiers and the number of subwindows processed for different grid spacing.

4.3.1 Description of the databases

The following face databases are used for this experiment:

CMU frontal face database: This database is usually known as CMU+MIT frontal face database, but we will refer to it as CMU frontal face database. The CMU frontal face database [90]

---

[1] The ground-truth information for all the face images are available publically from the website *http://vasc.ri.cmu.edu/idb/html/face/*

contains 117 images with 505 face images. Some example images from this database are shown in Figure 4.8.

CMU rotated face database: This database contains 50 gray-scale images with a total of 223 faces, of which 216 are rotated in the $[-90°; +90°]$ in-plane range. For our experiment, we will consider only faces within $[-90°; +90°]$ in-plane rotation range. Some example images from this database are shown in Figure 4.13.

CMU profile face database: The CMU profile face database [89] consists of 208 images with 441 faces of which 347 are profile views. Some example images from this database are shown in Figure 4.18.

4.3.2 Baseline frontal face classifier

The strong frontal face classifiers are trained with the same face database which was used for training a tree for location estimation (see Section 3.5.2 for the description of the cropped frontal face database). Additional to the face databases termed as "normal" (n) and "rotscale" (r), an additional one called "translation" (t) (where the face images are translated with +/-1 pixel translation in x and y direction) is introduced. The main idea of perturbing the cropped face database is to see the effect on the detection rate for different grid spacing. For each face image size (19x19, 19x27 and 24x24) and perturbation a strong classifier is trained. Let $C_F(X, Y)$ represent a baseline frontal face classifier, where X represents the face size that the classifier is trained with, and Y is the type of perturbation. The face size will be represented by only its height. For example, a strong classifier trained on faces with the size 19x27, and perturbation type "rotscale", will be represented as $C_F(27, r)$, and if a classifier needs to be represented only by its face size, then it will be represented as $C_F(27)$ by omitting Y.

Performance of the baseline classifier

Figure 4.4 shows the baseline performance of different classifiers on the CMU frontal face database. The baseline ROC curves are plotted by varying the classifier's threshold. The scanning grid spacing is fixed to 1, and the merging parameters for multiple detection are the same as mentioned in Section 4.2. and are fixed for the rest of this experiment. From the ROC curves, we observe that the performance is better for face image sizes of 24x24 and 19x27 than for the face image size of 19x19. We are also able to obtain a very good detection rate for reasonable number of false positives.

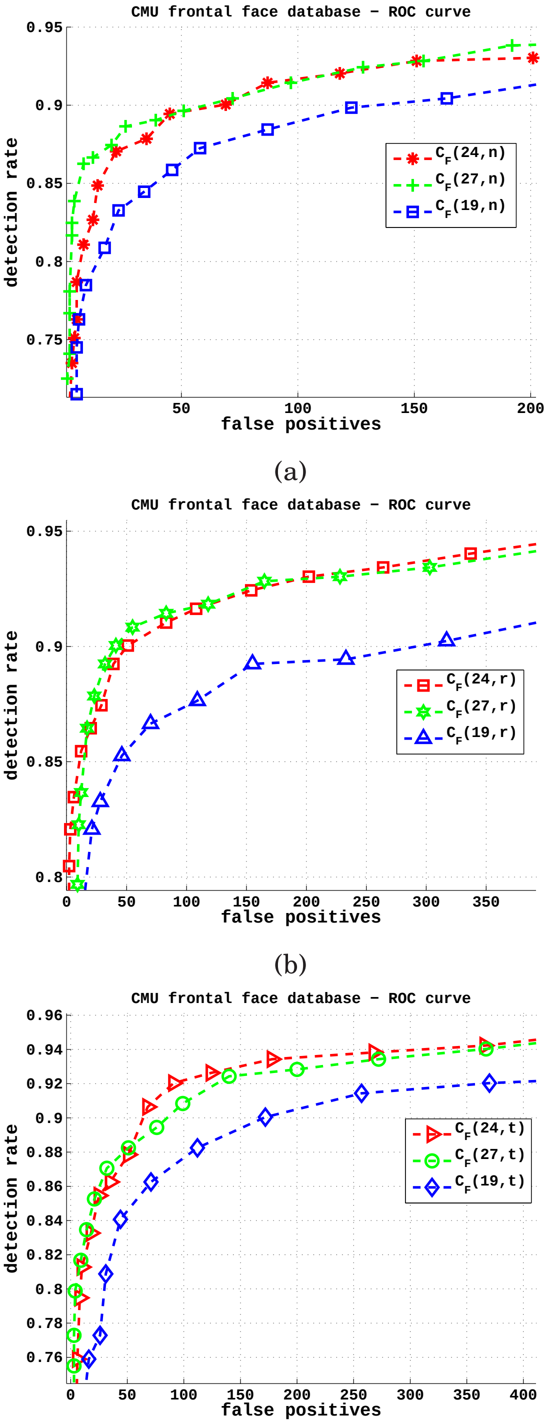
The detection rate of different classifiers with respect to grid spacing are also analysed and are shown in Figure 4.5. To obtain this plot, the threshold for each of the classifier is fixed such that the detection rate is 93% on the CMU frontal face database with scanning grid spacing of 1. We observe that the classifier with perturbation "translation" performs slightly better than the other two perturbations for different face image sizes and grid spacing. As the grid spacing is increased we can see that the detection rate drops drastically. For example, in Figure 4.5(a), the detection rate for the classifier $C_F(24, r)$ drops to 11.95% for the grid spacing of 14.

4.3.3 Results on frontal face detection

We now apply our proposed approach of using location estimation during the scanning process on the CMU frontal face database. The depth of the tree for location estimation is fixed to 15 for all different patch sizes. In the following paragraphs, we will discuss the effect on the detection rate with respect to the grid spacing, the effect of patch size on the detection rate for different grid spacing, and the computational time with respect to the detection rate obtained on this database.

Detection rate vs. grid spacing: Figure 4.6 shows the effect on the detection rate with respect to the grid spacing on the CMU frontal face database. From these figures, it is clearly visible that our approach obtains better detection rate when compared to its respective baseline for the same grid spacing, which validates our hypothesis experimentally (see Section 3.2 for description of our approach). For example, if we consider the baseline classifier $C_F(24, r)$, with the location estimator $T_F(16)$ in Figure 4.6(c), the detection rate obtained for grid spacing of 14 is 56.97%, which represents 45% relative improvement compared to the baseline alone, and only 35.86% less than the detection rate for the grid spacing of 2.

Effect of patch size on the detection rate: Again from Figure 4.6, we can observe the effect of different patch size used for location estimation on the detection rate for different grid spacing. All



(a)

(b)

(c)

Figure 4.4. Baseline ROC curves for different classifiers on the CMU frontal face database. (a) Classifiers trained without any perturbation ("normal"), (b) Classifiers trained with perturbation ("rotscale") and (c) Classifiers trained with perturbation ("translation").

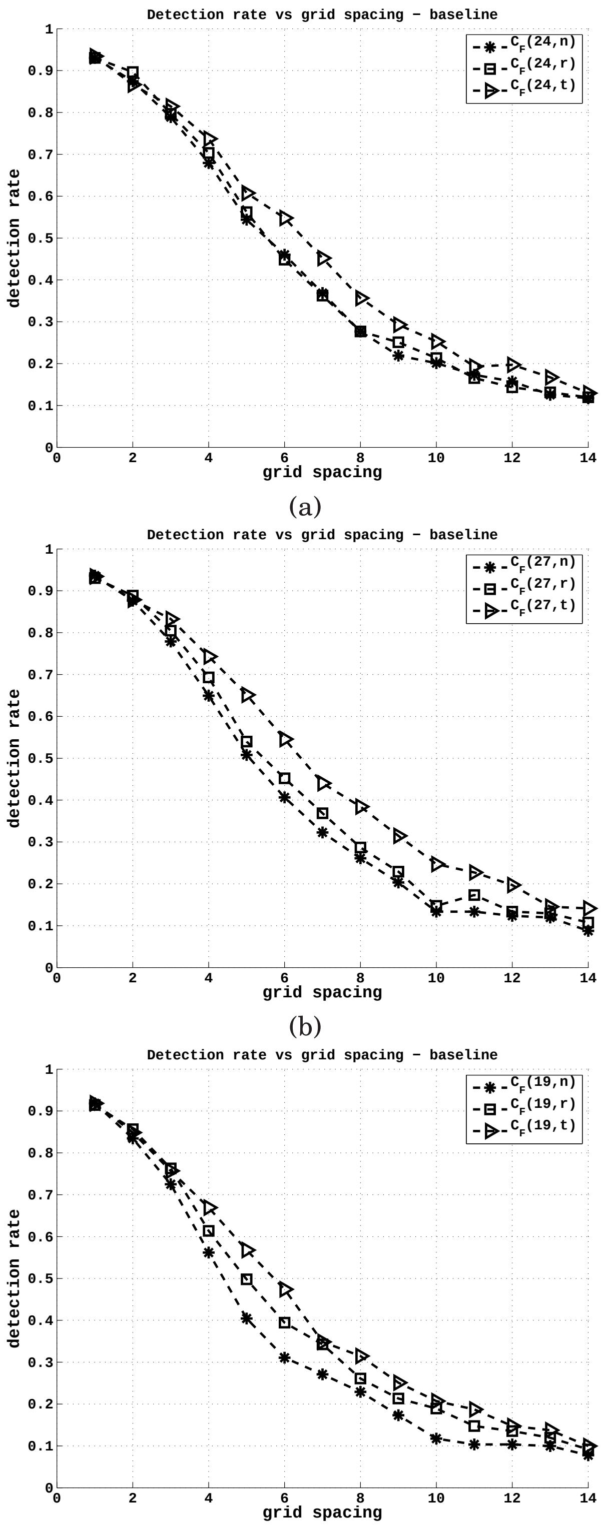
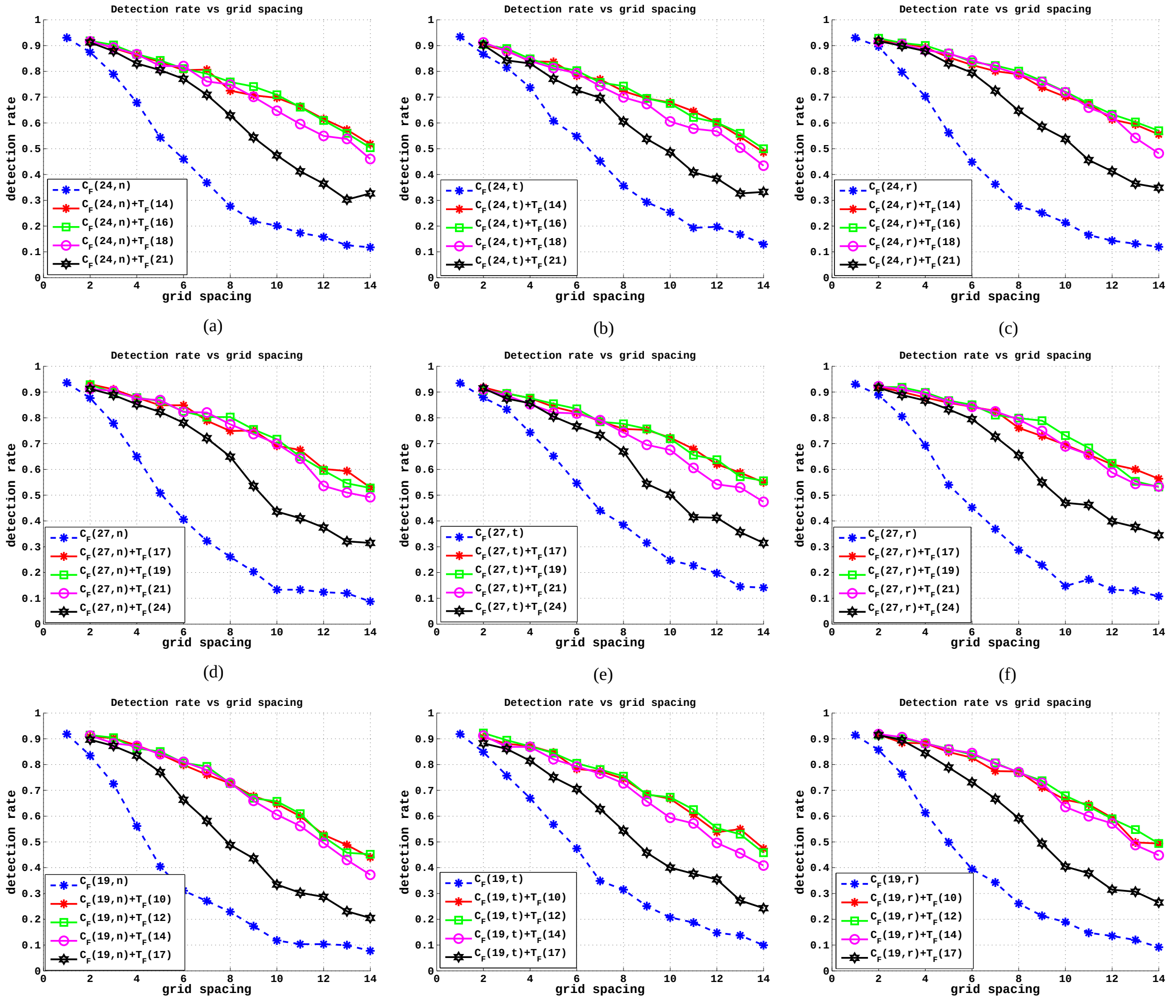


Figure 4.5. Baseline detection rate vs. grid spacing for different perturbation on the CMU frontal face database. (a) 24x24 face image, (b) 19x27 face image and (c) 19x19 face image.



(a) (b) (c)

(d) (e) (f)

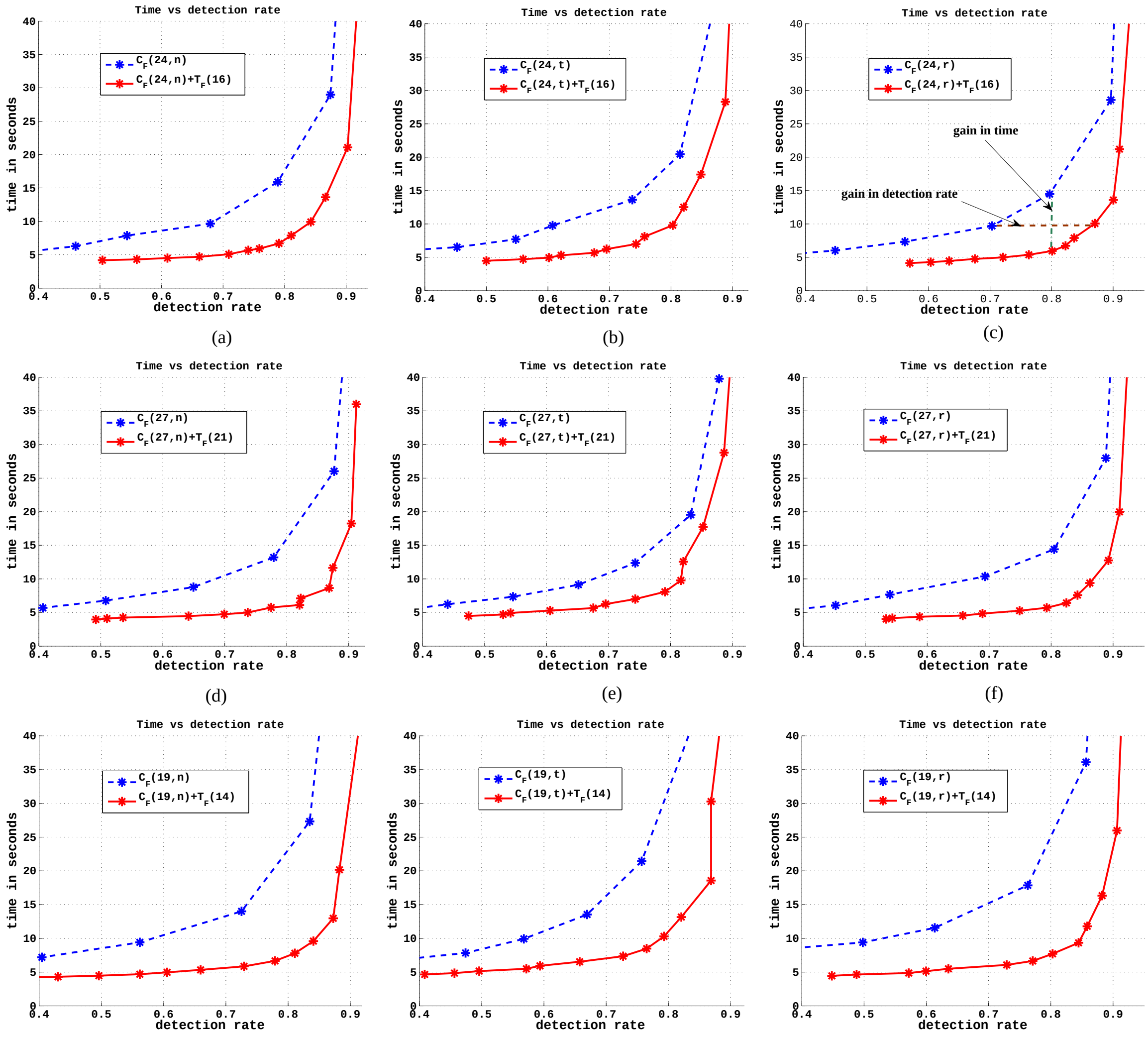
(g) (h) (i)

Figure 4.6. Detection rate vs. grid spacing with and without location estimation on the CMU frontal face database for different classifiers and face image sizes. (a,b,c) 24x24 face image, (d,e,f) 19x27 face image, and (g,h,i) 19x19 face image. The legend $T_F(X)$ represents a decision tree for location estimation and X represents the patch size.

the plots in Figure 4.6 follow a similar pattern. Let us take the Figure 4.6(c) for discussion. We can see that the performance can be clustered into two groups when the baseline classifier is combined with location estimation: One containing T_F trained with larger patch size of 21x21 (TL), while the other containing T_F trained with smaller patch sizes of 18x18, 16x16 and 14x14 (TS). We observe that TL performs similar to the groupTS upto a grid spacing of 4 and then starts to decrease more than TS. There is also not much difference in the detection rate between different patch sizes in TS. For the grid spacing of 14, there is almost a difference of 22.11% in the detection rate between TL and TS. This shows that smaller patch sizes for location estimation obtains better detection rate as the grid spacing is increased.

Computational time vs. detection rate: The computation time includes all the processing steps, such as scaling the image for the pyramid scan, calculation of integral images, and the application of a location estimator (only for our approach) and a baseline classifier. The computational time is analysed empirically. Figure 4.7 shows the computation time in seconds with respect to the detection rate for different classifier with and without location estimation. To reduce the clutter in a plot, we have selected one of the better performing baseline classifier with location estimation to compare with the performance of the baseline classifier. From these plots, we can see that our approach is faster than the baseline for a similar detection rate, and for a fixed computational time, our approach obtains a better detection rate. Note that our approach takes more computational time for the same grid spacing, but also can achieve better detection rate. For example, in Figure 4.7(c), our approach is 2.4 times faster than the baseline for the detection rate of 80%. If the computation time is fixed to around 10 seconds then a gain of 17% in the detection rate when compared to the baseline is achieved with our approach.

Some detection results are shown in Figure 4.8. We can observe that using location estimation reduces the number of miss detections with respect to the standard scanning technique for a larger grid spacing.

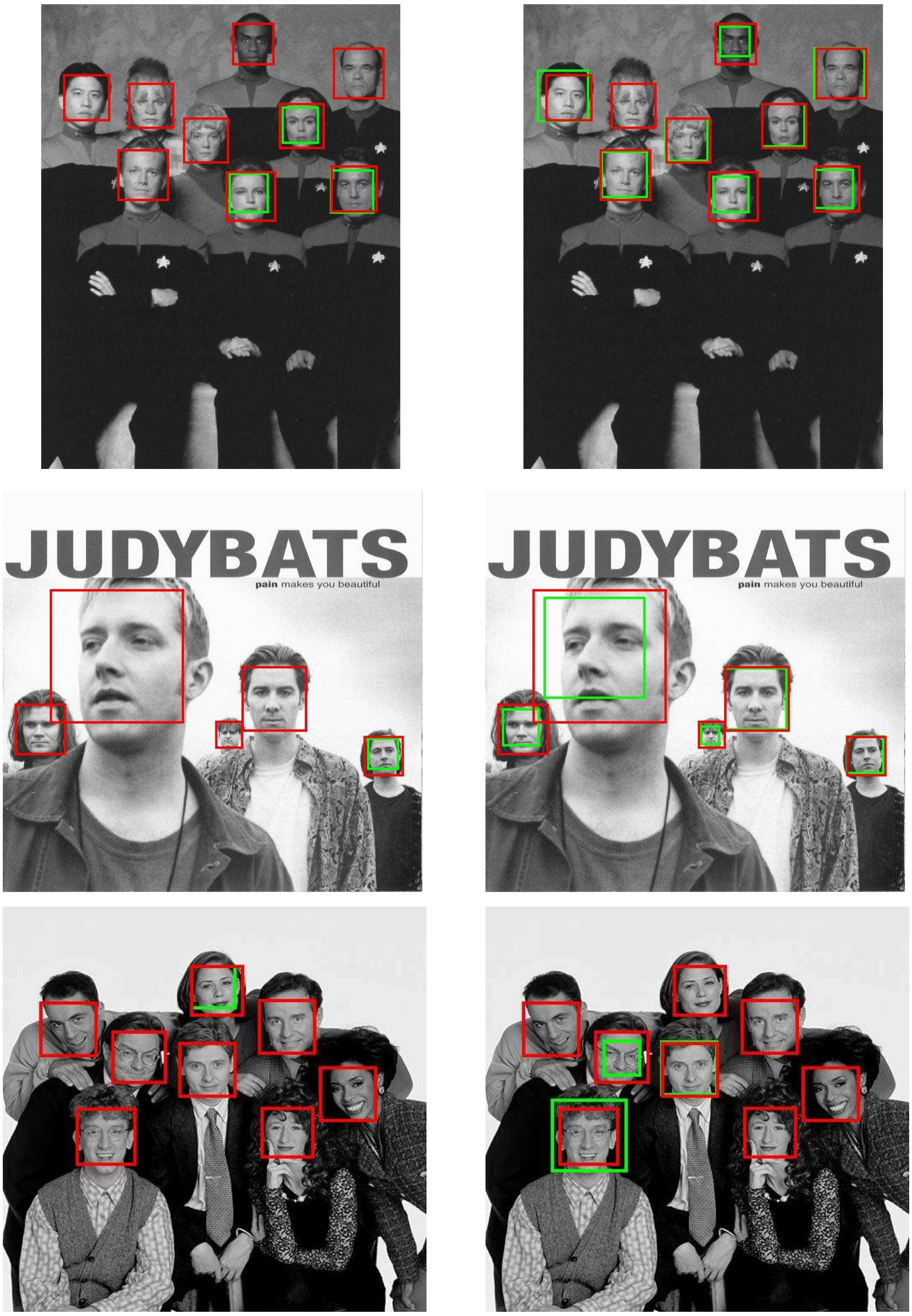


(a) (b) (c)

(d) (e) (f)

(a) (b) (c)

Figure 4.7. Detection rate vs. time with and without location estimation on the CMU frontal face database. The time is shown in seconds to scan all the 117 images from this database.(a,b,c) 24x24 face image, (d,e,f) 19x27 face image, and (g,h,i) 19x19 face image. The legend $T_F(X)$ represents a decision tree for location estimation and X represents the patch size.



(a) (b)

Figure 4.8. Some examples of frontal face detection on the CMU frontal face database. (a) with standard approach, and (b) with location estimation. The detection results are shown for the scanning grid spacing of 12. The red boxes represents the ground-truth and the green boxes represents detected faces.

4.3.4 Baseline frontal in-plane rotated face classifiers

The classifiers are trained with the cropped face database as described in Section 3.5.3 which is of size 24x24. A square face image size is considered for this task, since it is easier to handle in-plane rotation, and also we have seen before in Section 4.3.2 that the baseline classifier for face size of 24x24 performs better than 19x19. The baseline classifier for detecting frontal in-plane rotated faces consists of a rotation estimator and view-specific classifiers. The block diagram for our approach is shown in Figure 4.9. As described in Section 3.5.3, the rotation angles that we consider for our experiment ranges from -90° to +90° in steps of 10°, which results in 19 different views. For a view-specific classifier faces with +/-10° rotation angles are included to be slightly tolerant to errors from rotation estimator. Hence, 19 view-specific classifiers are trained as described in Section 4.2.3.

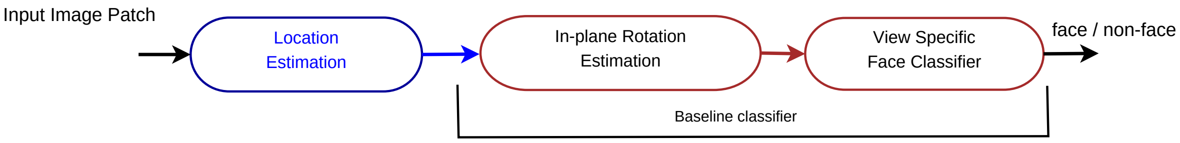


Figure 4.9. Overview of our approach for frontal in-plane rotated face detection. A patch smaller than the face size is used for face location estimation. Estimated location of the face is then passed through a rotation estimator and finally a view-specific face classifier is used to verify the presence of the face.

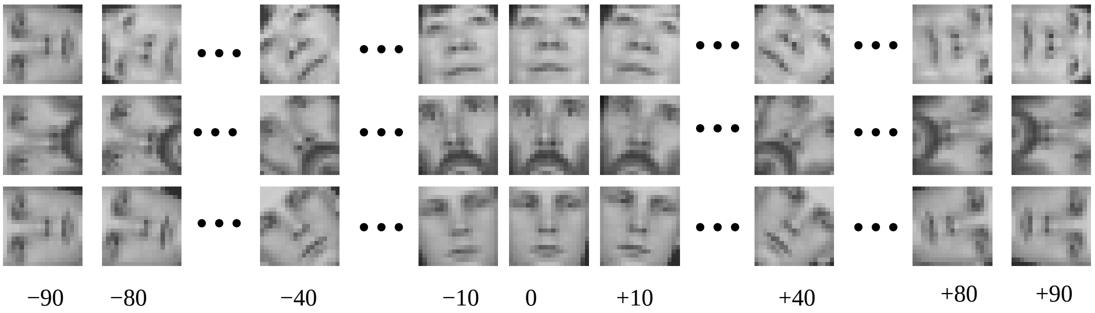


Figure 4.10. Patch examples used for training a rotation estimator. The rotation angle varies from -90° to +90° in step of 10°.

Next, we will look at the training and measuring the performance of an in-plane rotation estimator followed by the performance of the baseline classifier.

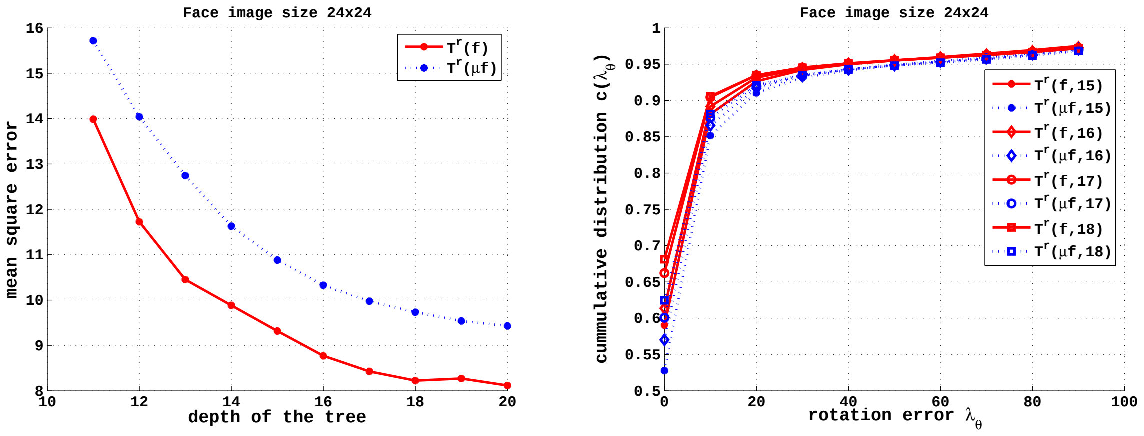
In-plane rotation estimator

The in-plane rotation estimator is built using a decision tree similar to the decision tree for location estimation as described in Section 3.4 (offset values are replaced by rotation angles). Now, each view of the patch is associated with a rotation angle and a decision tree is trained such that the leaf node learns and estimates the rotation angle for the corresponding patch appearances. Some examples of the rotated face samples are shown in Figure 4.10. Similar to the offset error defined in Equation 3.13, we define rotation error as

$$\lambda_\theta = |(\hat{\theta} - \theta)| \tag{4.4}$$

where $\hat{\theta}$ is the estimated and θ is the ground-truth rotation value. The number of test patches that have estimation error λ_θ is given by $g(\lambda_\theta) = \sum_{i=1}^{N} \mathbf{1}_{\lambda_\theta}(\lambda_\theta^{\mathbf{i}})$, where $\mathbf{1}_{\lambda_\theta}$ is the indicator function. The cumulative distribution is given by $c(\lambda_\theta) = \sum_{j=0}^{\lambda_\theta} g(j)$.

The tree for in-plane rotation estimator is represented as $T^r(B, D)$, where D represents the depth of the tree, and B represents the type of binary feature (μf or f). The face size is omitted, since here we are only considering one face size, which is 24x24. The trees are trained with different depths (11 to 20) and the performance of the rotation estimator for the binary tests $t_{\mu f}$ and t_f are shown in Figure 4.11. The tree $T^r(f)$ has lower MSE compared to the tree $T^r(\mu f)$ as seen from the Figure 4.11(a). We also observe from Figure 4.11(b) that, the rotation estimator can achieve above 85% of the test patches falling within an absolute rotation error of 10° for the different tree depths considered in this plot.



(a) (b)

Figure 4.11. Evaluation of in-plane rotation estimation. (a) mean square error vs. depth of the tree. (b) cumulative distribution with respect to the rotation error (λ_θ).

Performance of the baseline classifier

The baseline performance on the CMU rotated face database is shown in Figure 4.12(a). The rotation estimator $T^r(\mu f, 17)$ is used for the baseline classifier along with 19 view-specific classifiers and together it is represented as $C_R(24)$. The evaluation setup is the same as described in Section 4.2. We can see from the Figure 4.12(a) that the baseline classifier can achieve above 92% detection rate with couple of hundreds false positives.

4.3.5 Results on frontal in-plane rotated face detection

We now apply our proposed approach of using location estimation during the scanning process on the CMU frontal in-plane rotated face database. The depth of the tree for location estimation are fixed to 18 for all different patch sizes. In the following paragraphs, we will discuss the effect on the detection rate with respect to the grid spacing, the effect of patch size on the detection rate for different grid spacing, and the computational time with respect to the detection rate obtained on this database. To obtain the plots in Figure 4.12(b) and (c), the threshold of the baseline classifier is set at 94.4% detection rate on the ROC curve shown in Figure 4.12(a).

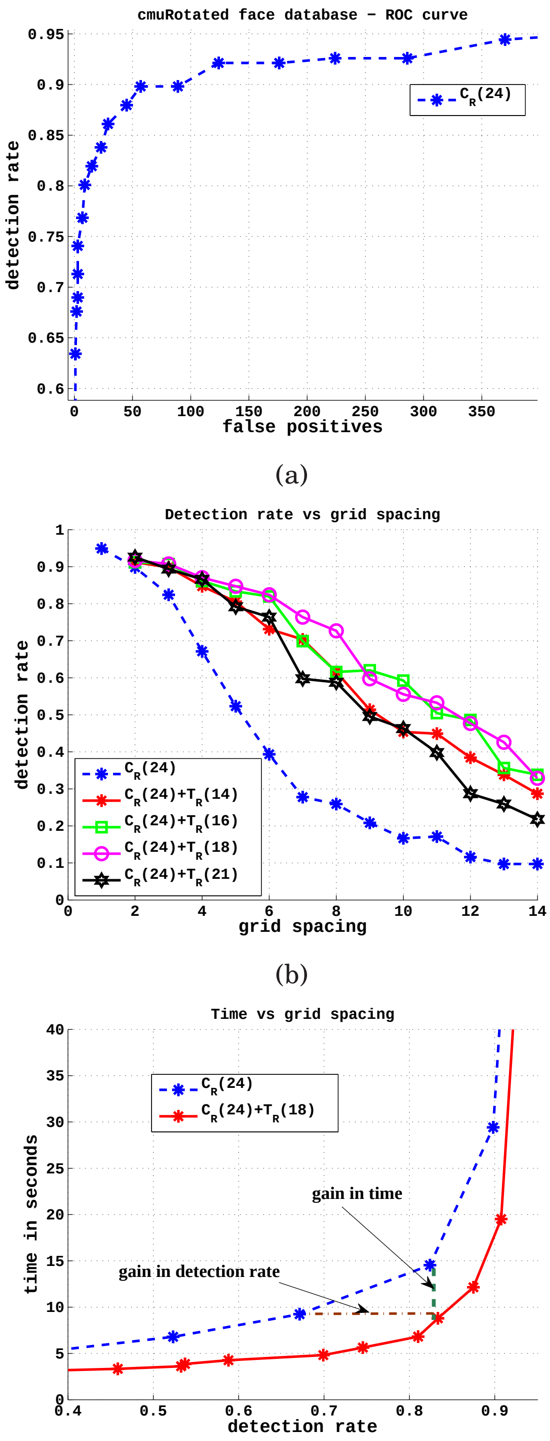
Detection rate vs. grid spacing Figure 4.12(b) shows the effect on the detection rate with respect to the grid spacing. We observe from the figure that with our approach better detection rates are achieved for larger grid spacing when compared to the baseline. For example, the baseline classifier obtains 9.72% detection rate for a grid spacing of 14, while with location estimation ($C_R(24) + T_R(18)$) the detection rate is 32.87%, resulting in a gain of 23.15%.

Effect of patch size on the detection rate: We refer again to Figure 4.12(b) to see the effect of the patch size on the detection rate with respect to the grid spacing. The effect of patch size on the detection rate in this case is not as clear as in frontal face detection (see Figure 4.6). But considering only the baseline classifier with $T_R(18)$ and $T_R(21)$, we can see that the smaller patch size performs better than the larger patch size used for location estimation.

Computational time vs. detection rate: Figure 4.12(c) shows the computation time in seconds with respect to the detection rate for the baseline classifier with and without location estimation.

Again, to reduce the clutter in a plot, a better performing classifier with location estimation ($T_R(18)$) is selected. From this plot, we can see that our approach is faster than the baseline for a fixed detection rate, and for a fixed computational time, our approach obtains a better detection rate. For example, in Figure 4.12(c), our approach is 1.7 times faster than the baseline for the detection rate of around 82.4%. If the computation time is fixed to around 10 seconds then there is a gain of 16% in the detection rate with our approach when compared to the baseline.

Some detection results are shown in Figure 4.13. We can again observe that, using location estimation reduces the number of miss detections with respect to the standard scanning technique for a larger grid spacing.

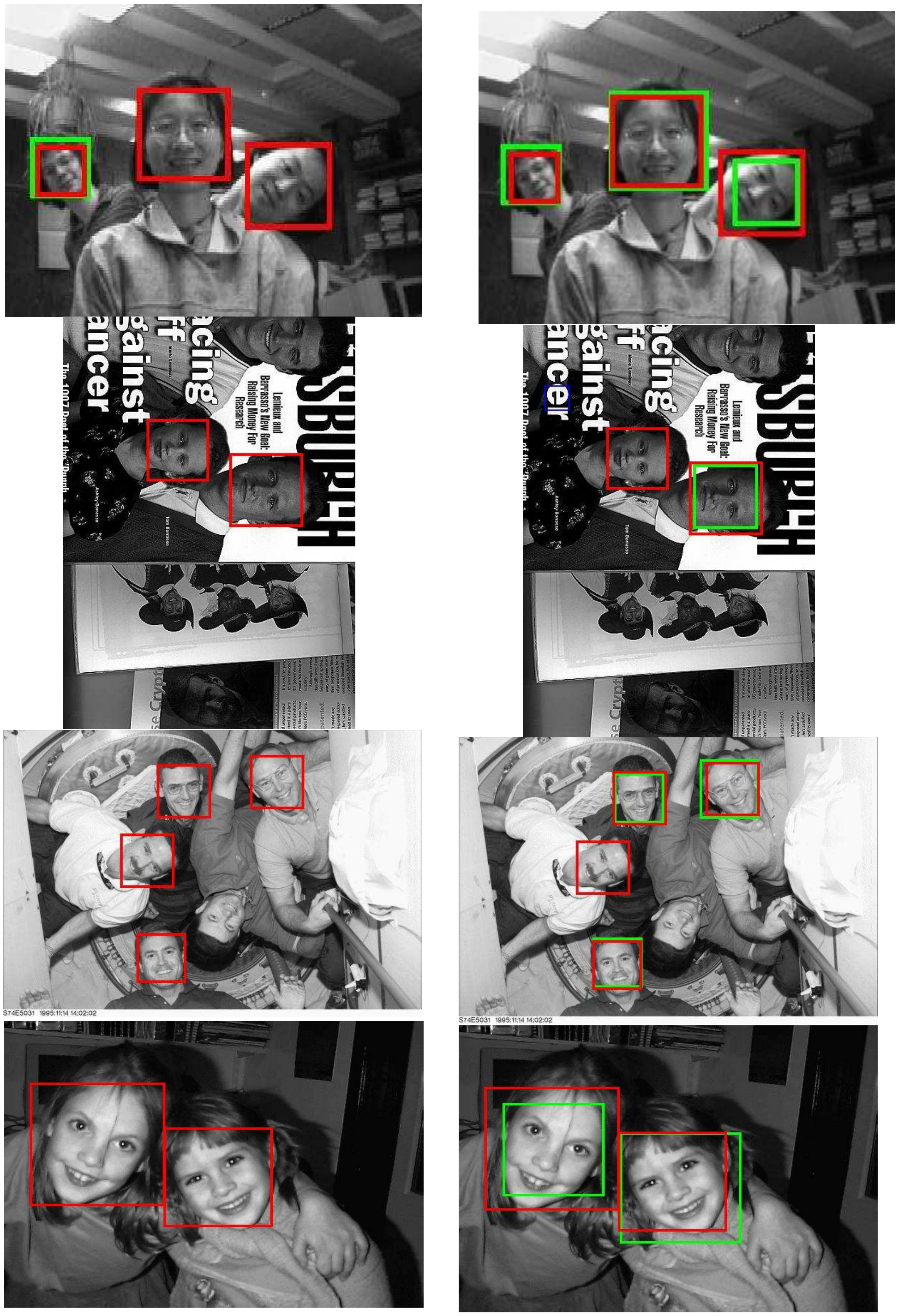


(a)

(b)

(c)

Figure 4.12. Performance on the CMU rotated face database. (a) shows the baseline ROC curve. (b) detection rate vs. grid spacing with and without location estimation. (c) detection rate vs. time with and without location estimation. Depth of the tree is fixed to 18. The legend $T_R(X)$ represents a decision tree for location estimation and X represents the patch size.



(a) (b)

Figure 4.13. Some examples of frontal in-plane rotated face detection on the CMU rotated face database. (a) with standard approach, and (b) with location estimation. The detection results are shown for the scanning grid spacing of 12. The red boxes represents the ground-truth, the green boxes represents detected faces, and the blue boxes represents false detections.

4.3.6 Baseline out-of-plane rotated face classifier

The baseline classifiers are trained with the cropped face database as described in Section 3.5.4. The baseline classifier for detecting out-of-plane rotated faces consists of an out-of-plane rotation estimator and view-specific classifiers. The block diagram for our approach is shown in Figure 4.14. The face image size considered here is of size 24x24. We grouped 67° and 90° face images as there was not much visual difference between them. With the grouping we obtain 3 views: frontal, left and right profile views. Three view-specific classifiers are trained using the same cropped face database. We will first look at the training and performance of an out-of-plane rotation estimator followed by the performance of the baseline classifier.

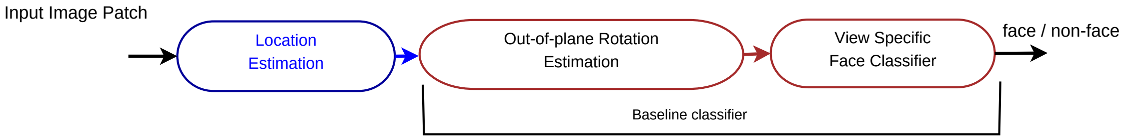


Figure 4.14. Overview of our approach for out-of-plane rotated face detection. A patch smaller than face size is used for face location estimation. Estimated location of the face is then passed through a out-of-plane rotation estimator and finally a view-specific face classifier is used to verify the presence of face.

Out-of-plane rotation estimator

Similar to the rotation estimator described in Section 4.3.4, a decision tree for estimating out-of-plane rotation is trained. Some examples of frontal and out-of-plane rotated face images are shown in Figure 4.15. The tree for out-of-plane rotation estimator is represented as $T^p(B, D)$, where D represents the depth of the tree, and B represents the type of binary feature (μf or f). Here too, we omitted the face size, since we are only considering one face size which is 24x24. Figure 4.16 shows the performance of an out-of-plane rotation estimator. We observe that the MSE for the decision tree $T^p(\mu f)$ is more than the decision tree $T^p(f)$, and that, as the depth increases, the MSE also slightly increases. The cumulative distribution shown in Figure 4.16(b) gives a better picture on the number of test patches that are correctly recognized. We see that nearly 92% of the test patches are correctly recognized without any errors, and there is not too much difference in performance with respect to the tree depth that are considered in this plot.

Performance of the baseline classifier

The ROC curve of the baseline classifier on the CMU profile face database is shown in Figure 4.17(a). The out-of-plane rotation estimator $T^p(\mu f, 14)$ is used for the baseline classifier along with 3 view-specific classifiers and together it is represented as $C_P(24)$. The evaluation setup is the same as described in Section 4.2, except for this database we had set α_l to 0.1 to consider overlapping of faces to some extent. We observe that it is difficult to obtain good performance in the case of out-of-plane rotated faces. The baseline classifier achieves above 60% with higher number of false positives compared to frontal and in-plane rotated baseline classifiers. Nevertheless, we will study the impact of location estimation on this baseline classifier.

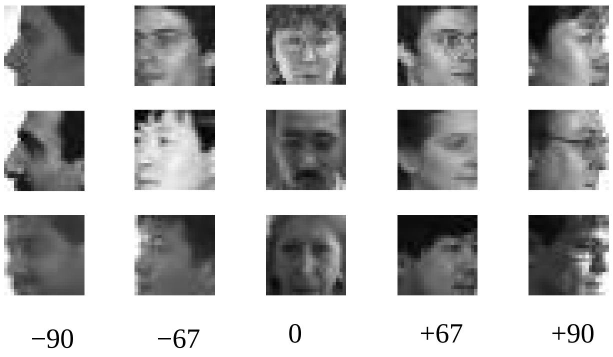
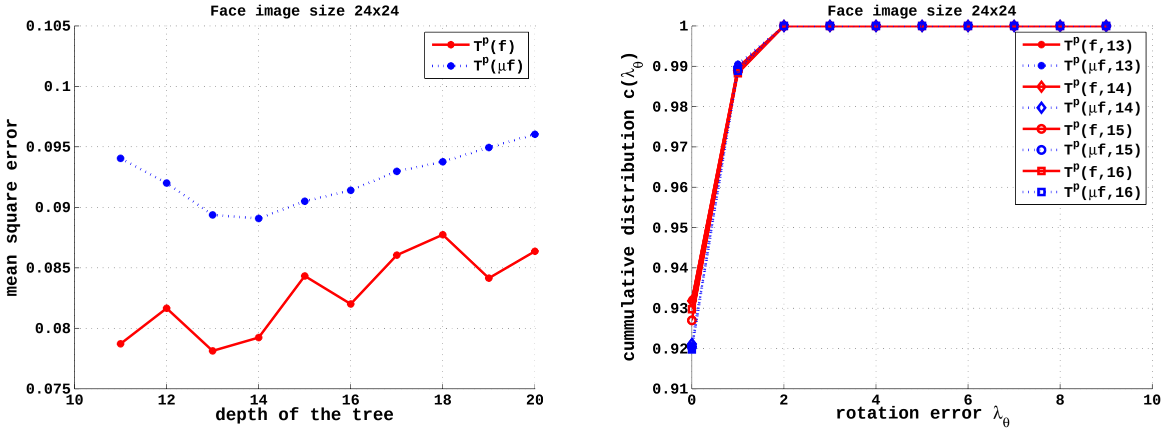


Figure 4.15. Patch examples used for training a profile estimator. The face image size is 24x24.



(a) (b)

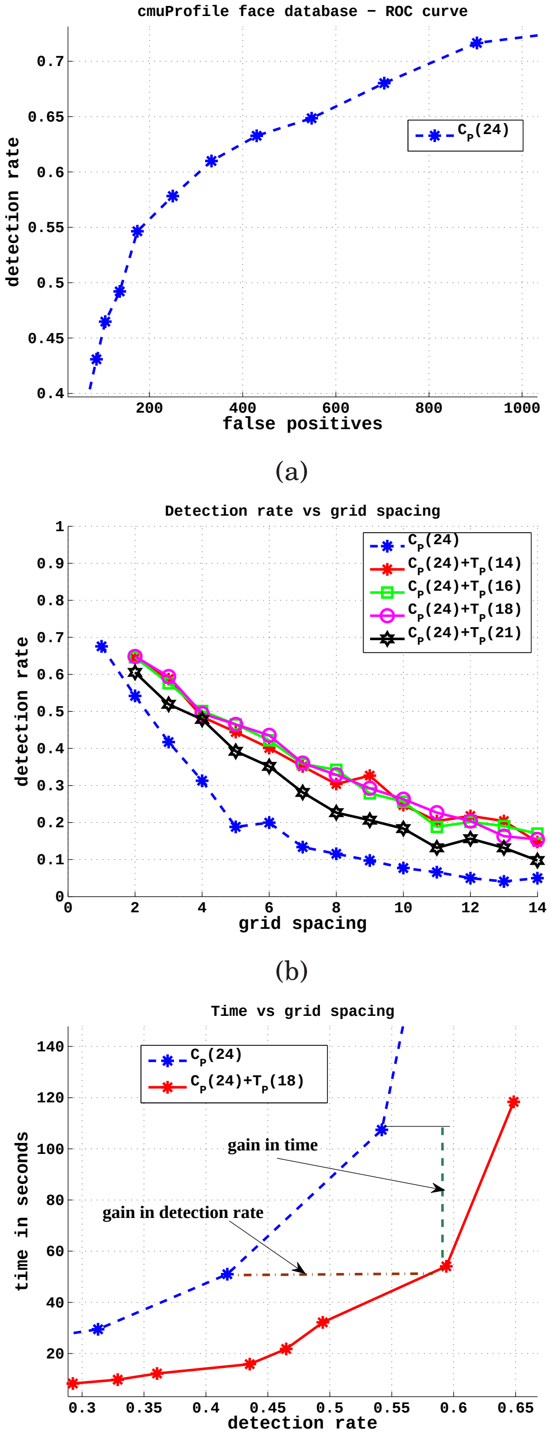
Figure 4.16. Evaluation of out-of-plane rotation estimation. (a) mean square error vs. depth of the tree. (b) cumulative distribution with respect to out-of-plane rotation error (λ_θ).

4.3.7 Results on out-of-plane rotated face detection

We now apply our proposed approach of using location estimation during the scanning process on the CMU profile face database. To obtain the plots in 4.17(b) and (c), the threshold of the baseline classifier is set at 67.7% detection rate on the ROC curve shown in Figure 4.17(a). Some detection results are shown in Figure 4.18. Here the detection results are shown for the grid spacing of 6. As before, we will discuss the effect on the detection rate with respect to the grid spacing, the effect of patch size on the detection rate for different grid spacing, and the computational time with respect to the detection rate obtained on this database.

Detection rate vs. grid spacing Figure 4.17(b) shows the effect on the detection rate with respect to the grid spacing. We observe from the figure that with our approach better detection rates are achieved for different grid spacings when compared to the baseline. For example, the baseline classifier obtains 5.0% detection rate for a grid spacing of 14, while with location estimation ($C_P(24) + T_P(18)$) it is 15.42%, resulting in a gain of 10.42%. But the advantage that we gain for this database is only minimal when compared to the frontal and in-plane rotated face detection.

Effect of patch size on the detection rate: We will refer again to Figure 4.17(b) to see the effect of the patch size on the detection rate with respect to the grid spacing. The effect of patch size on the detection performance is similar to the case of frontal face, and the patches could be clustered into two groups (TL consisting of patch size 21x21 and TS consisting of patch sizes 18x18, 16x16 and 14x14). But the difference in the detection rate between TL and TS is much lesser when compared to the experiments for frontal faces.



(a)

(b)

(b)

Figure 4.17. Performance on the CMU profile face database. (a) baseline ROC curve. (b) detection rate vs. grid spacing with and without location estimation. (c) time vs. detection rate. The legend $T_P(X)$ represents a decision tree for location estimation and X represents the patch size.

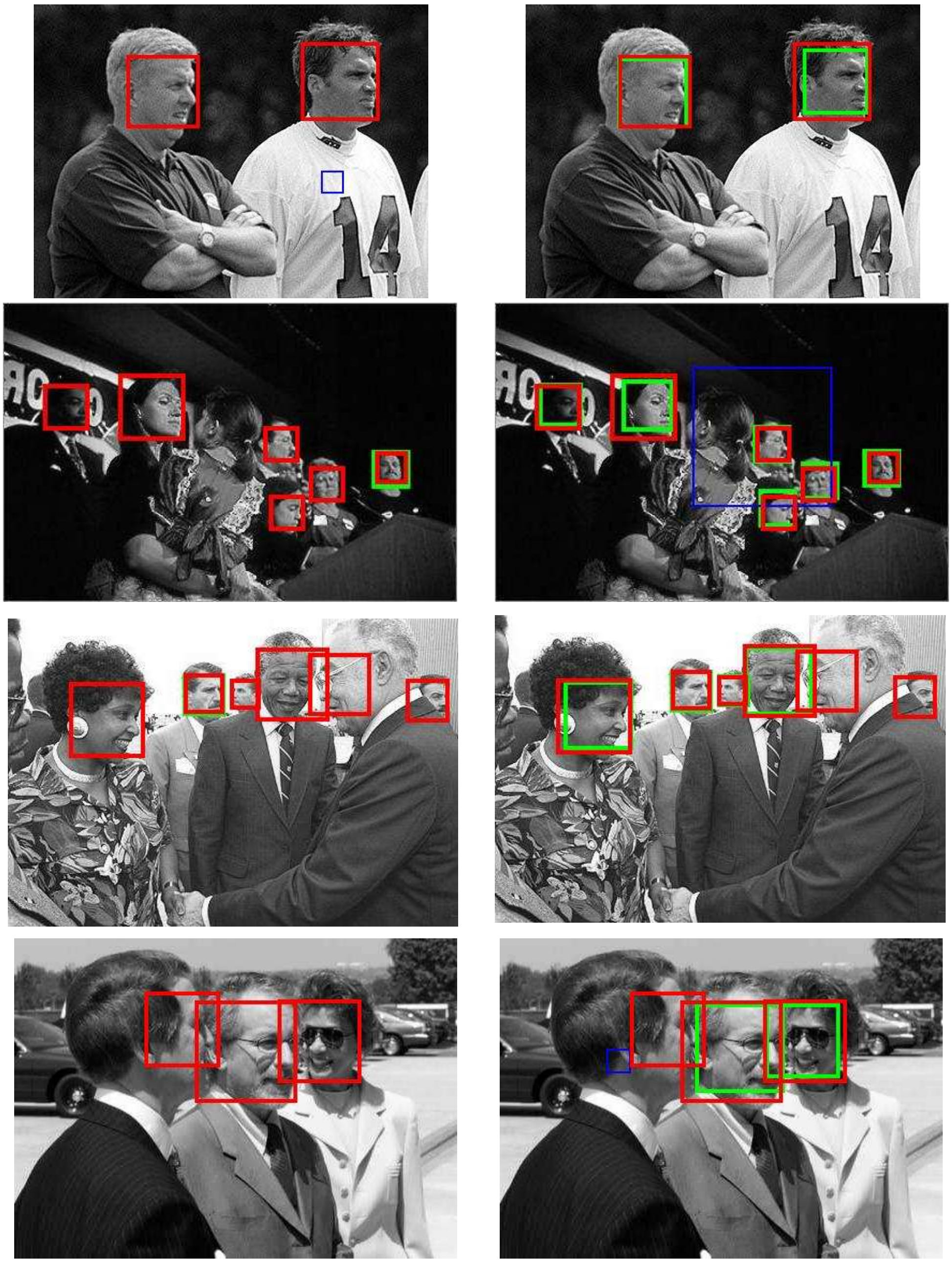

(a) (b)

Figure 4.18. Some examples of out-of-plane rotated face detection on the CMU profile face database. (a) with standard approach, and (b) with location estimation. The detection results are shown for the scanning grid spacing of 6. The red boxes represents the ground-truth, the green boxes represents detected faces, and the blue boxes represents false detections.

Computational time vs. detection rate: Figure 4.17(c) shows computation time in seconds with respect to the detection rate for the baseline classifier with and without location estimation. Again, to reduce the clutter in a plot, a better performing classifier with location estimation ($T_P(18)$) is selected. From this plot, we can see that our approach is definitely faster than the baseline for a fixed detection rate, and for a fixed computational time, our approach obtains a better detection rate. For example, in Figure 4.17(c), our approach is 2 times faster than the baseline for the detection rate of around 60%. If the computation time is fixed to around 50 seconds then there is a gain of 17.6% in the detection rate with our approach with respect to the baseline.

4.3.8 Discussion

We have shown that with our approach, we are able to achieve a better detection rate for a larger grid spacing when compared to the baseline, and also achieve a better computation time for a fixed detection rate. We will discuss in the following paragraphs, the computational time of the baseline classifiers, and the number of subwindows processed for different grid spacing.

Computational time of a strong classifier in different scenarios: The computation time is evaluated empirically. The scanning grid spacing was fixed to 2 and the total computation time of a strong classifier to process all the subwindows is noted. This value was then divided by the total number of subwindow processed to obtain the average processing time of the strong classifier for a subwindow.

We used the CMU frontal face database for evaluating the average processing time for frontal baseline classifier C_F. The computation time of a strong classifier in C_F takes around $1.4\mu Sec$ to process a subwindow without location estimation (T_F) activated, while it increases to $1.8\mu Sec$ when T_F is activated. The increase in processing time of a strong classifier can be due to the placing of a subwindow at more probable face like region. Therefore, the total processing time for one subwindow with location estimation would be $1.8\mu Sec$ plus the time to process T_F.

Similarly, for evaluating the average processing time for a strong view-specific classifier in C_R, we used the CMU rotated face database. A view-specific strong classifier in C_R takes around $1.5\mu Sec$ to process a subwindow without T^r and T_R activated, and $3.2\mu Sec$ when only T^r is activated. When both T^r and T_R are activated the processing time increases to $3.6\mu Sec$. Therefore, the total time to process a subwindow would be $3.6\mu Sec$ plus the time to process T^r and T_R.

For evaluating the average processing time for a strong view-specific classifier in C_P, we used the CMU profile face database. A strong view-specific classifier for the profile face, the computation time to process a subwindow is around $6.1\mu Sec$ without T^p and T_P activated, and $7.2\mu Sec$ with only T^p activated. When both T^p and T_P are activated the computational time increases to $8.1\mu Sec$ to process a subwindow. Therefore, the total processing time to process a subwindow would be $8.1\mu Sec$ plus the time to process T^p and T_P. The strong classifier for profile view takes 4 times more than the frontal face classifier for the same cascade architecture and feature set, indicating that the background does not get rejected as quickly as in frontal face classifier.

Number of subwindows processed for different grid spacing: To get an idea on the number of subwindows processed on the CMU frontal, rotated and profile face databases, we provide the data in the Table 4.1. The speed-up is shown with respect to the grid spacing of 1.

grid spacing	Number of subwindows processed			speed-up with respect to the grid spacing 1
	CMU frontal	CMU rotated	CMU profile	
1	73,533,029	32,927,327	50,238,423	1x
2	18,451,911	8,255,383	12,623,347	4x
3	8,222,911	3,679,183	5,645,619	9x
4	4,641,693	2,075,355	3,190,929	16x
5	2,984,850	1,332,412	2,060,841	25x
6	2,077,973	927,857	1,437,861	36x
7	1,529,509	683,066	1,059,434	49x
8	1,176,313	524,600	816,105	64x
9	932,856	415,583	648,801	81x
10	758,392	337,724	532,522	100x
11	628,880	279,813	440,026	121x
12	529,959	235,851	371,949	144x
13	409,157	201,471	318,145	169x
14	352,528	174,053	275,505	196x

Table 4.1. Total number of subwindows processed for the CMU face databases for different grid spacing (pyramid scaling factor 1.2).

4.4 Experiments on additional face databases

In this section, we will evaluate our approach on additional face databases such as Cinema and Web [27], FDDB [39], and CMU Multi-PIE [29].

4.4.1 Cinema and Web frontal face databases

Database description

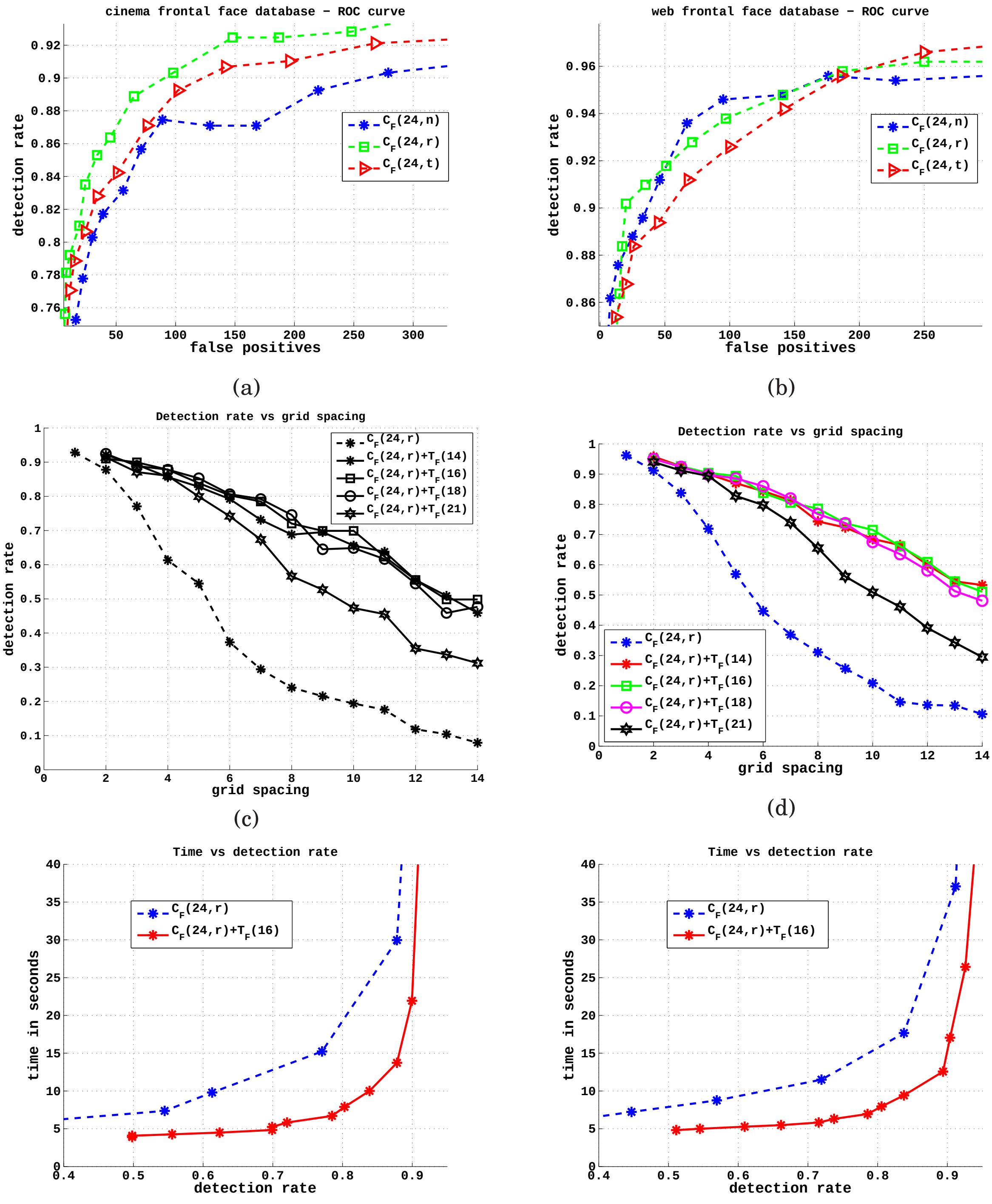
The Web face database contains 211 images with 499 faces, and the Cinema face database consists of 158 images with 276 faces. Some examples from this database along with detection results are shown in Figures 4.20 and 4.21. The results are shown for the grid spacing of 12.

Results

In this experiment, only one face size for the baseline classifier is considered, since we have already seen that the performance trend across different baseline classifiers are similar. We selected one of the better baseline classifier which is of face size 24x24 $C_F(24)$, based on the baseline ROC curves on the CMU frontal face database (Figure 4.6). The baseline ROC curves for the Cinema and the Web face databases are shown in Figures 4.19(a) and (b) respectively. The ROC curves are plotted for scanning grid spacing of 1. We can observe that the baseline classifiers are able to achieve very good detection rate for a couple of hundred false positives. Next we will look at the effect on the detection rate with respect to the grid spacing and also the computational time with respect the detection rate.

Figures 4.19(c) and (d) shows the effect on the detection rate with respect to the grid spacing with and without location estimation on the Cinema and the Web face database respectively. To obtain this plot, the baseline threshold is kept the same as used for the CMU frontal face database, and we have only shown for one of the baseline classifier $C_F(24, r)$. We observe similar behaviour as seen for CMU frontal face database (Figure 4.6). From the Figure 4.19(c) and (d), we can see that the detection rate for the grid spacing for 14 with location estimation is around 50% for both the databases, while the baseline is able to only achieve around 10%, thus a gain of 40% in detection rate is achieved with location estimation.

The computational time with respect to the detection rate for the Cinema and the Web face databases is shown in Figure 4.19(e) and (f) respectively. A similar observation of achieving a better detection rate for a fixed computational time or a better computational time for a fixed detection is seen for these databases.

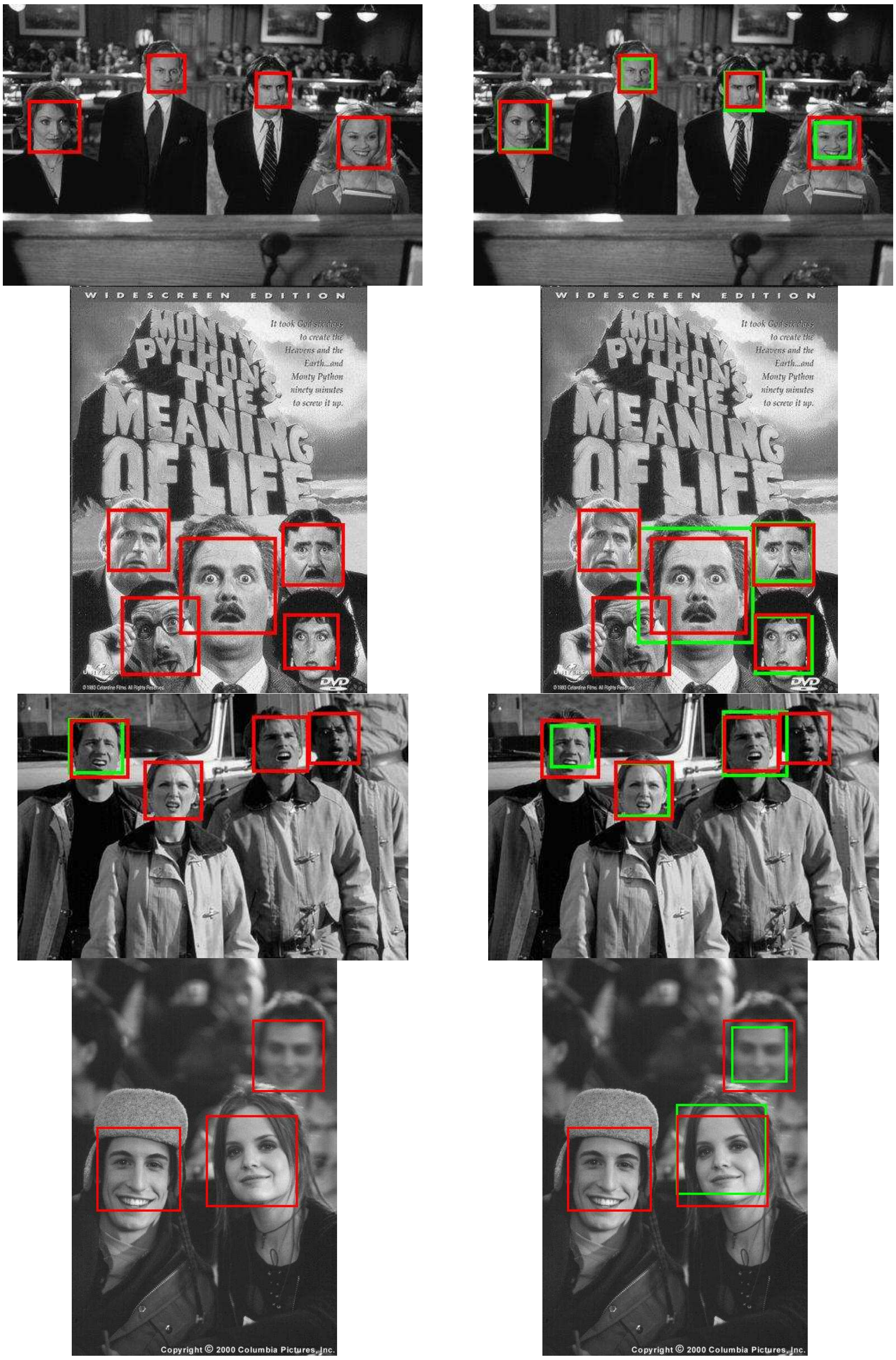


(a) (b)

(c) (d)

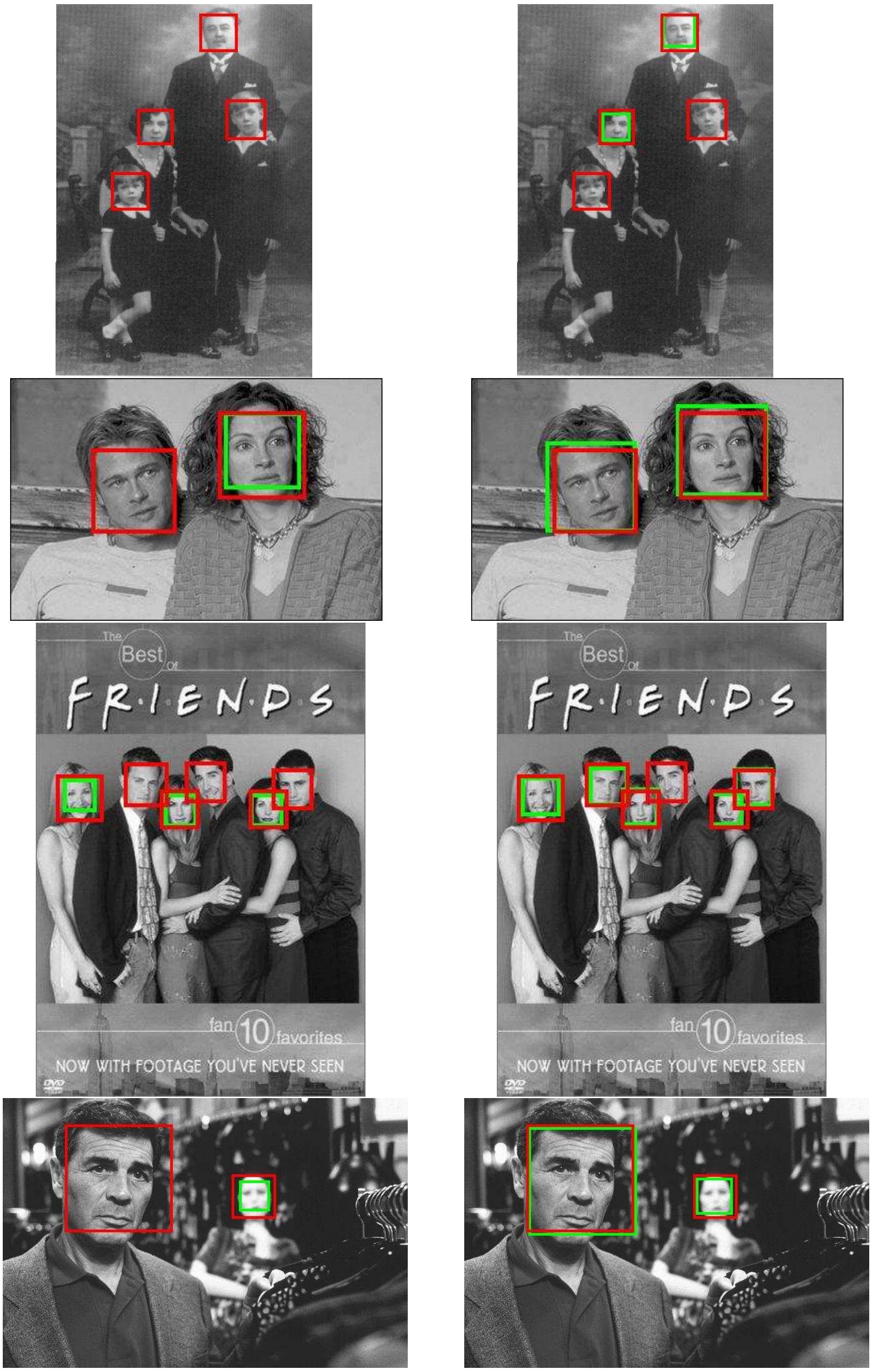
(e) (f)

Figure 4.19. Performance on Cinema and Web databases. Baseline ROC curve on (a) Cinema database, and (b) Web database. Detection rate vs. grid spacing for (c) Cinema, and (d) Web database. Detection rate vs. time for (e) Cinema, and (f) Web database.


WIDESCREEN EDITION
MONTY PYTHON'S THE MEANING OF LIFE
It took God six days to create the Heavens and the Earth...and Monty Python ninety minutes to screw it up.
DVD
Copyright © 2000 Columbia Pictures, Inc.

(a) (b)

Figure 4.20. Some examples of frontal face detection on the Cinema frontal face database. (a) with standard approach, and (b) with location estimation. The detection results are shown for the scanning grid spacing of 12. The red boxes represents the ground-truth and the green boxes represents detected faces.



(a) (b)

Figure 4.21. Some examples of frontal face detection on the Web frontal face database. (a) with standard approach, and (b) with location estimation. The detection results are shown for the scanning grid spacing of 12. The red boxes represents the ground-truth and the green boxes represents detected faces.

4.4.2 FDDB face database

Database description

The FDDB database was collected by Vidit Jain [39]. This database contains 2845 images with a total of 5171 faces. The author proposed a common evaluation framework to report the detection performance. There are two kind of experiments (EXP-1 and EXP-2) to report the results. The database is equally partitioned into 10 folds. In EXP-1, a 10-fold cross-validation is performed and the average of the 10 ROC curves is reported (9 folds are used for training a detector and 1 is used for validation). In EXP-2, the training database for the face classifier is unrestricted and the average of the 10 ROC curves obtained on 10-folds is reported. The author proposed to consider the problem of matching annotations and detections as finding a maximum weighted matching in a bipartite graph. The weight of the edge in the graph is based on the score value J (see Equation 4.1, overlap ratio of the two rectangles). The author also proposed a discrete score and a continuous score based on the value J. In discrete scoring the score is set to 1 if the value J is greater than 0.5, while for continuous scoring the value J is directly used for obtaining the correct and false detections.

Results

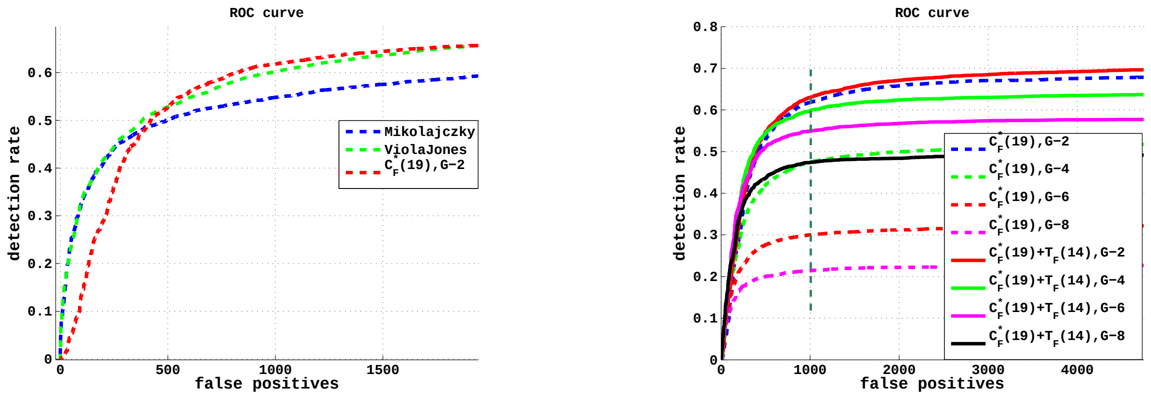
The results that we report here is for EXP-2 [2]. For this task we used an already trained detector from [86] which is based on modified LBP feature and will be represented as $C_F^*(19)$. The detector was trained on face size of 19x19. We use the standard evaluation toolbox from FDDB [3] to generate the ROC curves for the discrete score in Figure 4.22(a), and to compare with [107] and [64]. We can observe from this figure that the performance of our detector is comparable to the other two detectors.

Figure 4.22(b) shows the ROC curves with our detector using the standard scanning technique and with location estimation. For 1000 false positives, we can see that the detection rates with location estimation are higher than the standard scanning technique for respective grid spacing (the same is shown in Figure 4.23(a)). We can observe from the Figure 4.23(a) that the location estima-

---

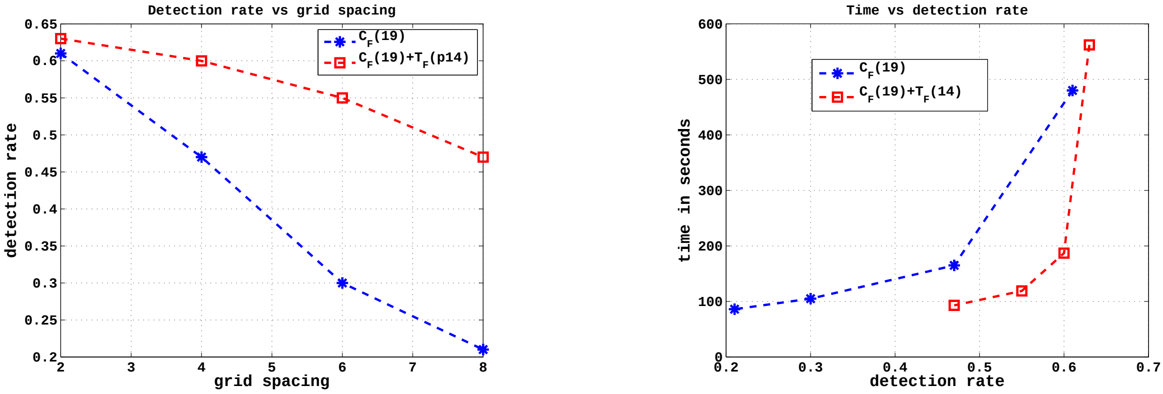
[2] These results were also reported in our paper [94], in the ECCV Workshop on Face Detection: Where we are and what next?.

[3] *http://vis-www.cs.umass.edu/fddb/results.html*



(a) (b)

Figure 4.22. ROC curves for the FDDB face database. (best viewed in color) (a) Comparison of our detector with (107) and (64). (b) ROC curves for our detector with standard scanning and with location prediction. The legend G-* represent grid spacing used while scanning.

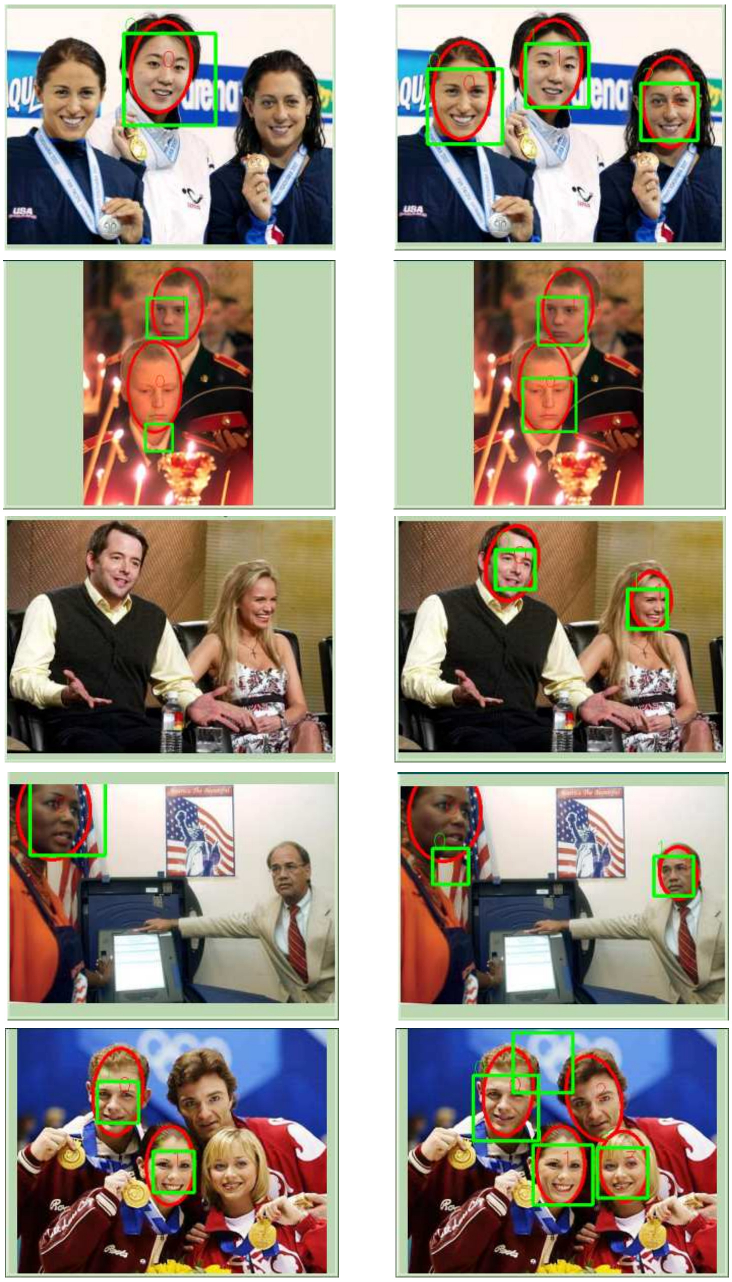


(a) (b)

Figure 4.23. Performance on the FDDB face database. (a) detection rate vs. grid spacing with and without location estimation, and (b) time vs. detection rate with and without location estimation.

tion with grid spacing 4 performs comparable to standard scanning technique with grid spacing 2, and location estimation with grid spacing 8 performs comparable to standard scanning technique with grid spacing 4.

The time taken to process 2845 images using standard scanning technique with grid spacing 2, 4, 6, and 8 are 480, 165, 105, and 86 seconds respectively, while with location estimation they are 562, 187, 119, and 93 seconds respectively. This is shown in Figure 4.23(b). For the detection rate of around 60%, our approach is 2.5 times faster, and for a detection rate of 47% our approach is 1.7 times faster than the standard scanning approach. Some detection results form this database are shown in Figure 4.24, for a grid spacing of 8, with and without location estimation.



(a) (b)

Figure 4.24. Some examples of frontal face detection on the FDDB face database (a) with standard approach, and (b) with location estimation. The detection results are shown for the scanning grid spacing of 8. The red ellipses represents the ground-truth and the green boxes represents detected faces. Note, the ground-truth is shown only if there is a match.

4.4.3 CMU Multi-PIE database

Database description

The CMU Multi-PIE database contains 337 subjects, imaged under 15 view points and 19 illumination conditions in up to four recording sessions [29]. A subset of images from this database is selected, for our experiment. We considered left profile (camera label 110), right profile (camera label 240) and frontal (camera label 051) face images from this database. All the different subjects with the recording session 01 and image number 00 for the above three views were considered, resulting in 249 images for each view. Each image contains only one face and it is more or less centred.

Results

For this experiment, we considered the baseline classifier $C_P(24)$ and the location estimator $T_P(18)$. The merging parameters are kept the same as for the experiments in Section 4.3.6. The ROC curves for the left profile, right profile and for the frontal face are reported in Figure 4.25. We observe that the left and right profile faces have slightly higher detection rate compared to the frontal face images. It should be noted that when we used the frontal baseline classifier $C_F(24)$ to detect frontal faces from this database, the detection rate was 100% with just 1 false detection, which indicates the influence of the out-of-plane rotation estimation on the detection performance on frontal faces.

Figure 4.26 shows the effect on the detection rate with respect the grid spacing and the computational time with respect to the detection rate for each of the face views. To obtain these plots, the threshold of the baseline classifier is fixed to 87% detection rate on the frontal ROC curve. Again, from the Figure 4.26(a,b,c), we observe that, a better detection rate is achieved for increasing grid spacing with location estimation when compared to the baseline, and from the Figure 4.26(d,e,f), we can see that, a better computational time could be achieved for a fixed detection rate or vice versa.

Some detection results for the left profile, frontal and right profile views are shown in Figures 4.27, 4.28 and 4.29 respectively. The results are shown for a grid spacing of 6. Some profile face images as shown in the first column of the Figures 4.27 and 4.29 are covered with hair and are difficult to detect.

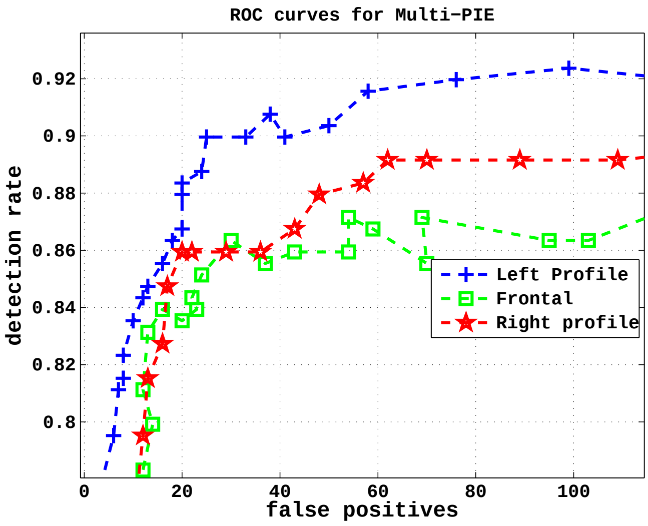
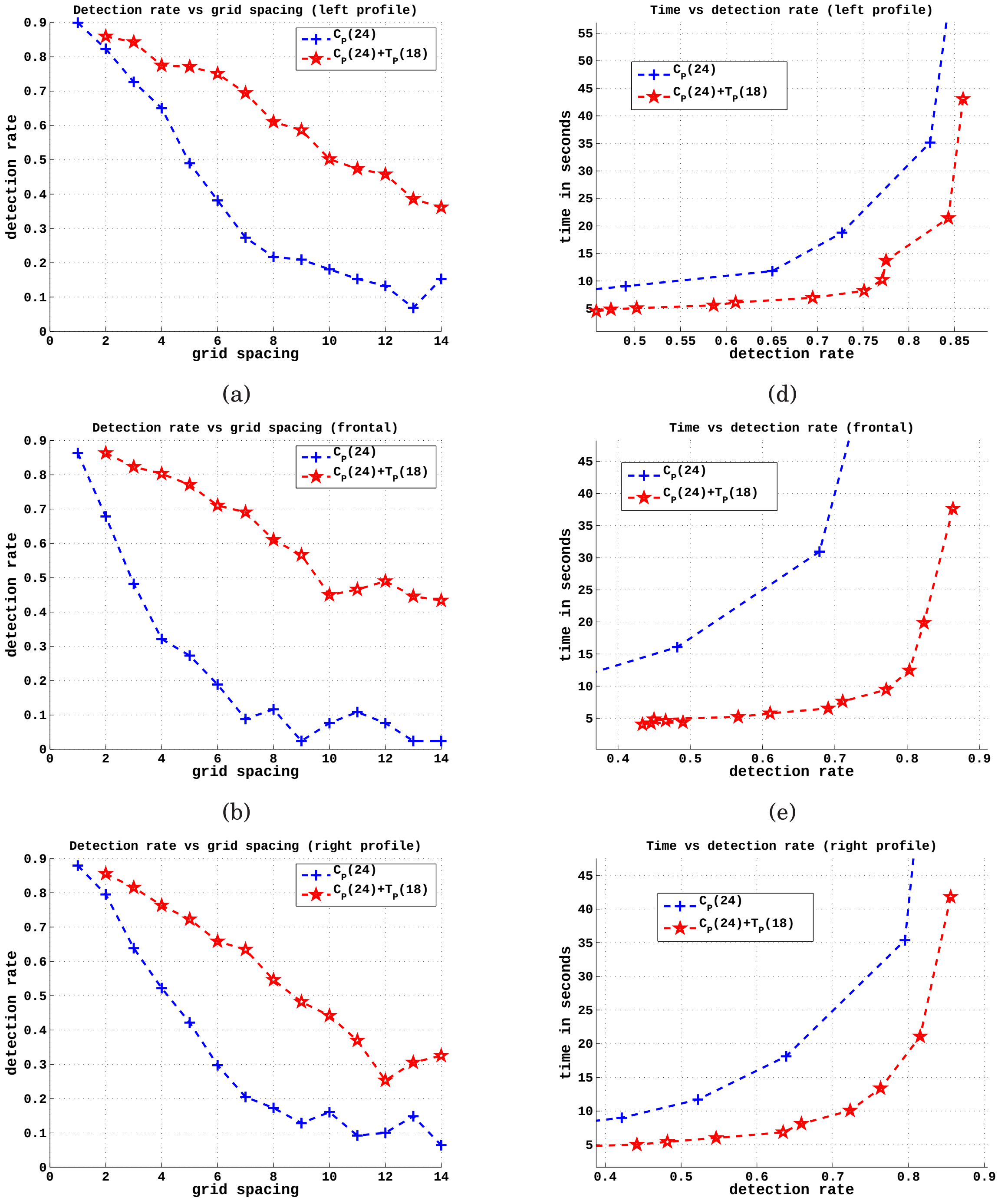


Figure 4.25. ROC curve on the Multi-PIE database



(a) (d)

(b) (e)

(c) (f)

Figure 4.26. Performance on the Multi-PIE database. (a,b,c) detection rate vs. grid spacing with and without location estimation, and (d,e,f) time vs. detection rate with and without location estimation.

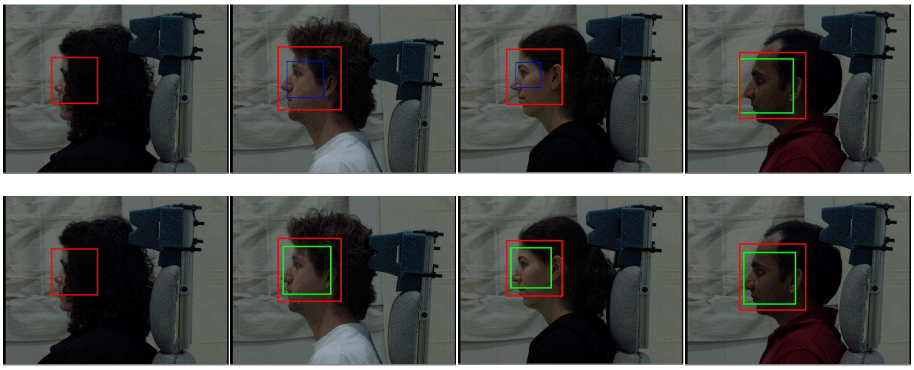


Figure 4.27. Some examples of left profile detection on the Multi-PIE face database. The first row shows the detection results with standard technique. The second shows the detection results with location estimation. The results shown here are for the scanning grid spacing of 6. The red boxes represents the ground-truth, the green boxes represents detected faces, and the blue boxes represents false detections.

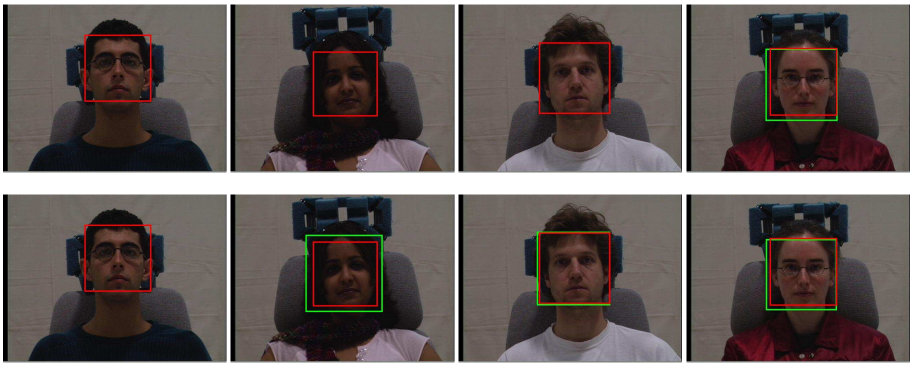


Figure 4.28. Some examples of frontal face detection on the Multi-PIE face database. The first row shows the detection results with standard technique. The second shows the detection results with location estimation. The results shown here are for the scanning grid spacing of 6.

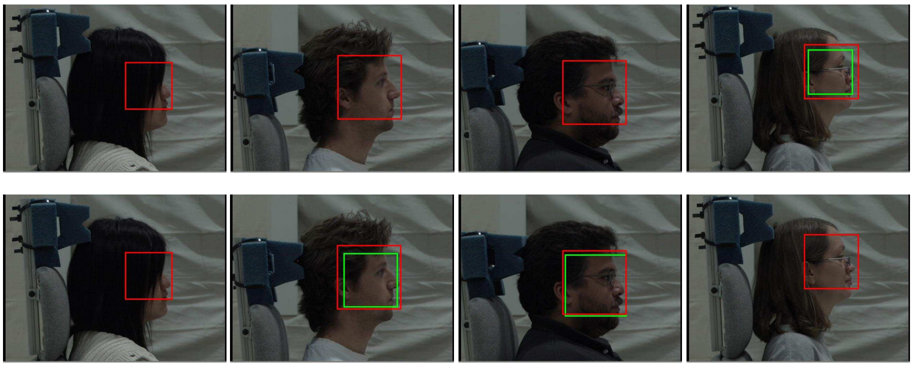


Figure 4.29. Some examples of right profile detection on the Multi-PIE face database. The top row shows the detection results with standard technique. The second shows the detection results with location estimation. The results shown here are for the scanning grid spacing of 6.

4.5 Summary

In this chapter we applied our proposed search technique on various benchmark face databases and showed that our approach indeed reduces the number of miss detection for larger grid spacing in the sliding window framework. We also trained decision trees for estimating in-plane and out-of-plane rotations, and have seen that they perform reasonably well. We have seen that for frontal face databases it is relatively easier to obtain good detection rate for larger grid spacing compared to in-plane rotated and profile face databases.

Chapter 5

On improving the proposed search technique using interest points

In Chapter 3, a search technique to reduce the number of miss detections, when lesser number of subwindows are processed was proposed. In the previous chapter, the proposed approach was validated on various face databases. To further improve this search technique, in this chapter, we investigate the use of interest point detection along with location estimation to focus on face like region for detection. The detected interest points could be considered as a non-regular grid as opposed to the regular grid previously considered.

Object detection using interest point detection and image patch descriptors have been popular for many different objects. Interest points are regions in the image which are consistent for different views and perturbation in the image (Gaussian [58], maximally stable extrema region (MSER) [19], Harris [31], and FAST [80]). Usually, a patch around an interest point is extracted and matched against the stored patches or a learned codebook [106; 80]. Many such patches around the interest points are extracted for an object, which could then be used for detection tasks [49; 65].

In the task of face detection, Hoogenboom et al. [33] proposed a method to find interest points to speed up the detection process. The authors observed that due to illumination the nose region was brighter than its neighbours, and therefore detect such region as an interest point. Yali Amit at al. [3] searched for spatial arrangements of edge fragments (center of two eyes and mouth) for focusing on face like region. In [30], eye regions were extracted based on its property that it looks darker than the rest of the face, and can be compared to MSER interest point extraction. Yow et al. [118] proposed pre-attentive feature selection based on derivative filters to detect facial features such as the eyes, nose and the mouth region from an image. In this chapter, we will also use the gradient information for interest point detection and then use location estimation to increase the chances of detecting the faces in an image.

This chapter is organized as follows. In the following section, an overview of our approach is presented. Section 5.2, describes our proposed interest point detector. In Section 5.3, experiment results on different face databases are reported. Finally, summary is presented in Section 5.5.

5.1 Improving the proposed search technique

In the sliding window framework the location estimator was placed on a regular grid. Our goal here is to replace the regular grid by a grid which is based on some features from the image that helps to focus more on the target region. Such a grid could be obtained by using an interest point detector. Some examples of interest points detected on the images containing faces are shown in Figure 5.2. We can see that there are many interest points lying within the face region.

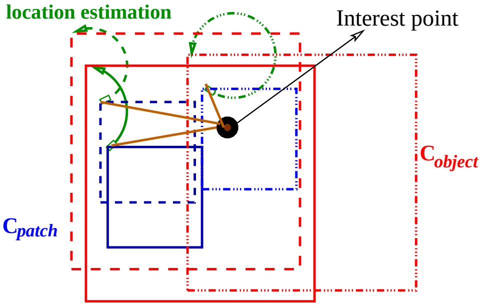


Figure 5.1. Illustration of our face detection approach for one interest point. We sample some image patches around an interest point, and apply the location estimator (C_{patch}) on the patches to further estimate the face position. The blue boxes are the sampled image patches around an interest point. The red boxes are the estimated face location by the location estimator.

Then for a given interest point, some image patch regions are sampled around it, and the face position is estimated using a location estimator for each sampled image patch. Figure 5.1 shows an overview of our approach for one interest point. The sampling of the image patches around an interest point should be such that at least one image patch falls completely within the face region for the location estimator to estimate the face location correctly. If the interest points are mainly located in some parts of the face region (such as the eyes, the nose, or the mouth), then image patches could be sampled assuming that the interest point can be any of these locations. The reason of applying a location estimator is to increase the chances of locating the face region for any small errors in the detected location of the interest point.

To detect faces with different scales, we follow a pyramid based approach as before and instead of recalculating the interest point at every scale, we scale the location of interest points. In the next section, we describe a simple interest point detector based on quantized gradient orientation.

5.2 Interest point detection

The proposed interest point detector is based on quantized gradient orientation [16] which is simple and also fast to compute. Given an image I the gradient in x (g_x) and y (g_y) direction is computed by:

$$g_x(i,j) = I(i,j-1) - I(i,j+1) \tag{5.1}$$

$$g_y(i,j) = I(i-1,j) - I(i+1,j) \tag{5.2}$$

where i and j represents image index. The magnitude (M) and the orientation (O) are given by:

$$M(i,j) = \sqrt{g_x(i,j)^2 + g_y(i,j)^2} \tag{5.3}$$

$$O(i,j) = tan^{-1}\frac{g_y(i,j)}{g_x(i,j)} \tag{5.4}$$

Next, a binary image (B) based on the gradient orientation O is created, which is given by:

$$B(i,j) = \begin{cases} 1 & \text{if } O(i,j) \in [\pi_l, \pi_u] \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

where π_l and π_u represents the lower and upper gradient orientation values. Given the binary image B, connected component labelling [56] is applied to find different regions. The centroid of each region forms our interest point. Figure 5.2(b) shows the interest points detected using the

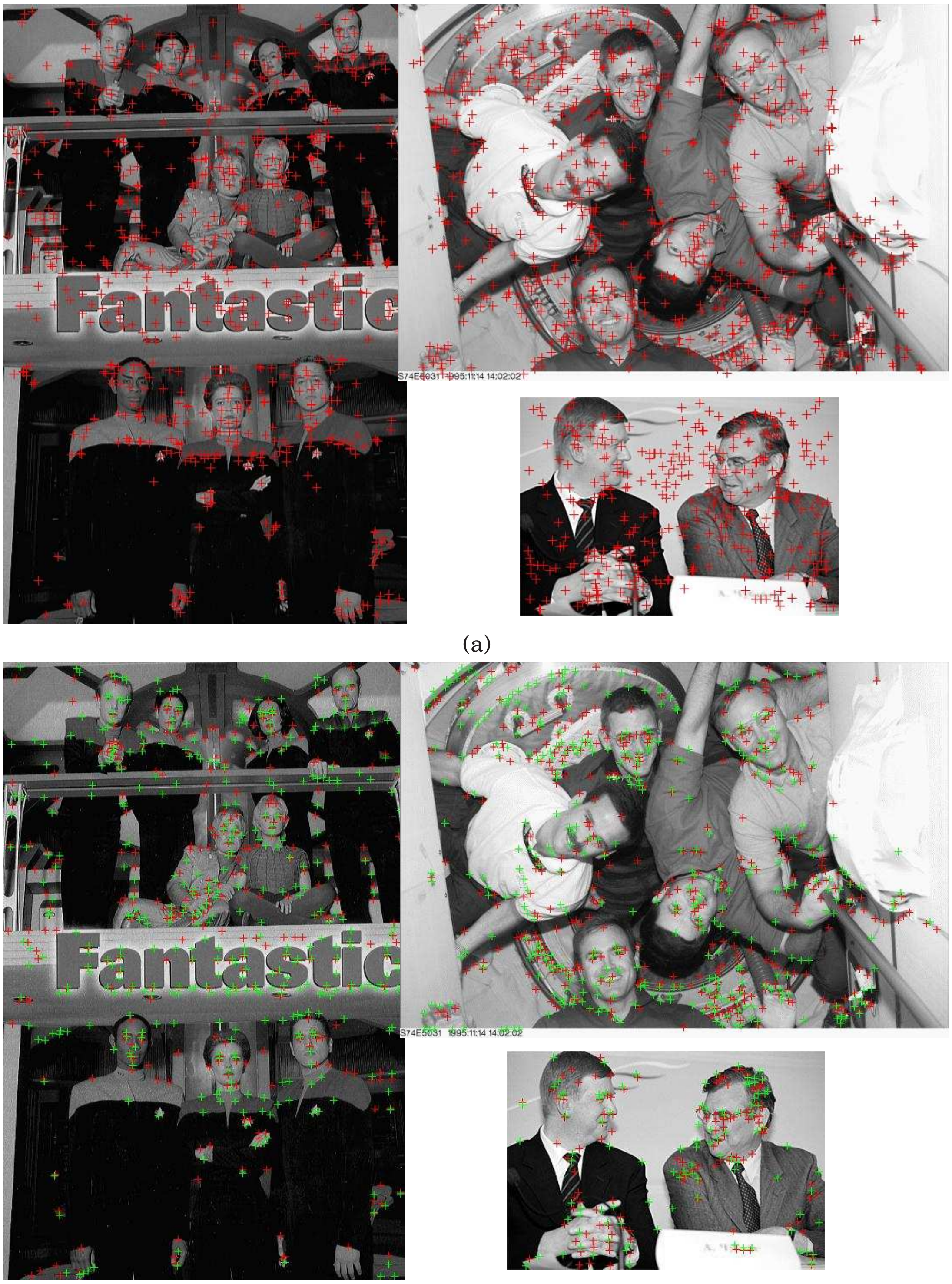


Figure 5.2. Examples of interest point detector on the images containing faces. (a) using difference of Gaussian based detector, (b) our proposed interest point detector based on quantized gradient orientation. The two cross colors represents interest point obtained for two different gradient orientation quantization $(45°, 135°)$(green color) and $(-45°, -135°)$(red color).

proposed approach. In this figure, interest points detected for two different quantized gradient orientations are shown in two different colors (red and green). Each color represents the interest points obtained by a different quantization of gradient orientation (π_l and π_u).

Two parameters control the number of interest point generated by our approach. One is the threshold on the gradient magnitude M, and the other is the threshold on the region area in the binary image B. While creating the binary image B, gradient orientation O can be considered only if the magnitude is above a specified threshold (m_t). The number of regions in the binary image could also be reduced by considering those regions that have at least a minimum number of pixels (r_t) within the region.

5.3 Experiments

We now apply the proposed approach on different face databases and compare with the sliding window approach with and without location estimation. First, we describe the parameters used for interest point detection, and then the sampling of image patches around an interest point for frontal and profile faces. Finally, the results on the CMU face databases are reported.

5.3.1 Experiment setup

The parameters required for the interest point detection are mainly, gradient orientation quantization $[\pi_l, \pi_u]$, gradient magnitude threshold m_t, and minimum number of pixels in a binary region r_t.

Gradient orientation quantization $[\pi_l, \pi_u]$: In [118], the authors have observed that at low resolution, the 6 facial features (left and right eyebrow, left and right eye, nose and mouth) appear only as dark elongated blobs against the light background of the face. We also use this knowledge and set the values of $[\pi_l, \pi_u]$ to $[45°, 135°]$ and $[-45°, -135°]$, which captures the gradient orientations in positive and negative vertical direction (y axis) targeting mainly the eye, mouth and nose region.

Gradient magnitude and region area threshold: The values of gradient magnitude threshold m_t and region area threshold r_t controls the number of interest points detected within the face region. To analyse the effect of varying the values of m_t and r_t, we conduct a small experiment on the CMU frontal face database. For each value of m_t and r_t, we count the total number of interest points detected in all the images and the number of faces having zero interest point that fall within its bounding box which is shown in Table 5.1. To visualize the table a surface plot of the Table 5.1 is shown in Figure 5.3. We can see from the table that as the threshold value is increased, more number of faces have no interest point falling within its bounding box. The values of m_t and r_t needs to be set such that the total number of interest points in the image are reduced while all faces have at least one interest point lying within its bounding box. There can be different combination of m_t and r_t (shown in bold numbers in Table 5.1, and marked in ellipse in Figure 5.3) that can satisfy the above criteria. Since we are considering to detect faces as small as 24x24, a smaller value of r_t and larger value of m_t may be possibly preferable. A very small value of r_t also increases the number of interest point detected in the image. Assuming that for a face image of 24x24 there are at least 6 pixels having the gradient orientation between $[45°, 135°]$ or $[-45°, -135°]$, we set r_t to 6. Once r_t is fixed, we have only 2 or 3 values of m_t to be selected from the table, and any one could be selected. We selected a lower threshold value m_t=28 for the rest of the experiments.

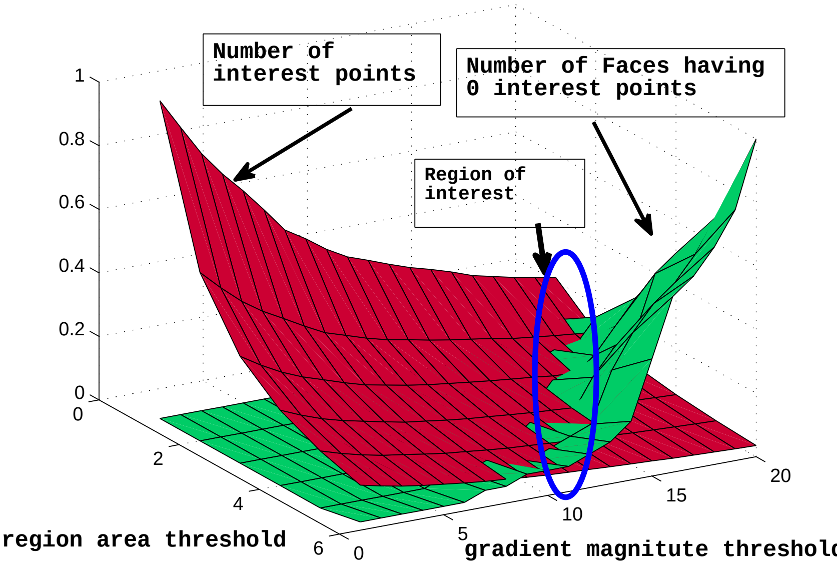


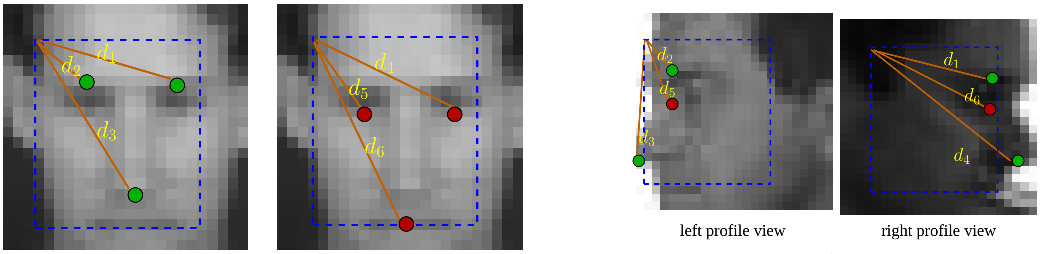
Figure 5.3. Surface plot of table 5.1.

r_t m_t	2	4	6	8	10	12
2	0(708732)	0(375621)	0(238666)	0(168200)	0(127073)	1(100628)
4	0(640538)	0(333012)	0(210350)	0(148601)	0(112777)	1(90099)
6	0(573501)	0(294206)	0(185805)	0(132022)	0(101052)	1(81067)
8	0(519399)	0(263270)	0(166500)	0(119101)	0(91599)	**1(73973)**
10	0(473256)	0(238446)	0(151577)	0(109138)	0(84376)	**1(68289)**
12	0(422215)	0(213094)	0(136463)	0(99009)	**0(76850)**	**1(62418)**
14	0(368687)	0(189330)	0(122764)	1(89675)	**1(70023)**	2(57054)
16	0(340352)	0(174905)	0(114000)	1(83508)	**1(65316)**	2(53304)
18	0(308122)	0(159979)	0(104783)	1(76748)	2(60065)	3(49000)
20	0(282637)	0(147701)	0(97235)	0(71425)	1(55854)	3(45671)
22	0(266279)	0(138879)	0(91466)	**0(67339)**	1(52639)	3(43046)
24	0(248934)	0(129827)	0(85481)	**0(63021)**	2(49342)	4(40453)
26	0(232944)	0(121513)	**1(79967)**	**2(58960)**	3(46213)	5(37837)
28	0(219918)	**0(114632)**	**0(75402)**	1(55603)	4(43681)	7(35778)
30	**0(205855)**	**0(107270)**	**0(70636)**	2(52191)	10(40955)	14(33521)
32	**1(188932)**	**1(98795)**	3(65195)	6(48174)	14(37791)	21(30956)
34	**2(178237)**	2(93133)	3(61708)	9(45597)	17(35851)	23(29375)
36	3(167676)	3(87680)	5(58139)	10(43059)	19(33864)	26(27706)
38	3(158794)	3(82888)	8(54943)	17(40644)	22(31896)	30(26073)
40	3(150406)	6(78384)	11(52157)	19(38610)	26(30274)	38(24758)

Table 5.1. Total number of interest points detected (values inside the bracket) and faces having 0 interest points (value outside the bracket) lying within its bounding box on the CMU frontal face database for varying gradient magnitude threshold and region area threshold. The numbers in bold show the likely m_t and r_t values, which could be selected for interest point detection.

Sampling of the image patches around an interest point: The sampling of image patches around an interest point is based on the knowledge that the interest points would be detected near the eye, nose and mouth region. The face image size that we considered here is of size 24x24.

Figure 5.4 shows roughly the location of the interest points considered for the frontal and out-of-plane rotated faces. The offset values for the frontal in-plane rotated faces are kept the same as for the frontal faces. Let each offset be represented by a number from 1 to 6 as shown in the Figure 5.4. The number of image patches around an interest point can be controlled by selecting only a subset of the offset values from d_1 to d_6. Let $\mathcal{O}_{1:X}$ represent the number of offset values considered, where X can take values from 1 to 6. In our experiment, we consider $\mathcal{O}_{1:6}$, $\mathcal{O}_{1:5}$, $\mathcal{O}_{1:4}$, $\mathcal{O}_{1:3}$, $\mathcal{O}_{1:2}$ and $\mathcal{O}_{1:1}$, to vary the number of image patches around an interest point.



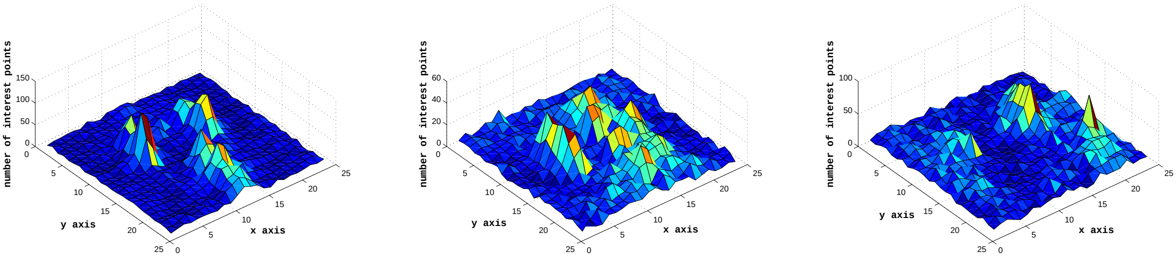
left profile view

right profile view

(a)

(b)

Figure 5.4. Sampling of the image patches assuming that the interest points are located near the eye, nose and the mouth region. (a) Illustration for a frontal face image, and (b) Illustration for a profile face image. d_1,d_2, d_3, d_4, d_5, d_6 represents the 6 offset values considered in our experiments. The red and green dot represents interest point detected for different gradient orientation quantization.



(a)

(b)

(c)

Figure 5.5. Histogram of interest point for the CMU face databases. (a) For the CMU frontal face database, (b) for the CMU rotated face database, and (c) for the CMU profile face database.

We also show the histogram of interest point on the CMU frontal, rotated and profile face databases in Figure 5.5. To obtain the histogram, we first detect the interest points and then based on the ground-truth face bounding box, we scale the location of the interest points and populate a

24x24 matrix. For the CMU frontal face database, the histogram peaks at the eye and mouth nose region and is clearly seen in Figure 5.5(a). In the case of CMU rotated face database, the interest points are spread around the eye and mouth-nose region as seen from the Figure 5.5(b). Ideally, if there are equal number of faces for each rotation angle, then the histogram would be evenly distributed, but it is not so in the CMU rotated face database. This is because, there are around 135 faces out of 216 that have -30° in-plane rotation. The CMU profile face database contains around 289 right profile and 75 left profile faces, and hence the histogram peaks are near the eye and the nose region of the right profile face as seen in Figure 5.5(c).

5.3.2 Evaluation on the CMU face databases

In this section, the results for the CMU frontal, rotated and profile face databases are reported, and compared with the results obtained in Chapter 4. The evaluation setup is kept the same as described in Section 4.2 (scaling factor to build the image pyramid and merging parameters). We use the same baseline classifier and location estimator as used in Chapter 4 to be consistent for comparison. When using interest points for face detection, the legend in a figure will be represented as “IP”.

Tables 5.2, 5.3, and 5.4 shows the detection rate, false detections, and the computational time in seconds for different number of subwindows processed for the CMU frontal, rotated and profile face database respectively. The number of subwindows processed in Tables 5.2, 5.3, and 5.4, are comparable to the number of subwindows processed with regular grid spacing approximately from 6 to 14 in Table 4.1.

Number of sub-windows tested	detection rate	time in seconds	number of false detections
1,548,331 ($\mathcal{O}_{1:6}$)	91.4%	8.6	41
1,345,508 ($\mathcal{O}_{1:5}$)	90.6%	8.0	38
1,141,865 ($\mathcal{O}_{1:4}$)	89.6%	7.5	31
932,084 ($\mathcal{O}_{1:3}$)	88.8%	6.9	29
636,164 ($\mathcal{O}_{1:2}$)	86.0%	6.2	22
329,626 ($\mathcal{O}_{1:1}$)	78.6%	5.3	11

Table 5.2. Evaluation of the detection rate, speed and the number of false detections using interest point with different number of image patches on the CMU frontal face database. $\mathcal{O}_{1:X}$ represents the number of image patches considered around an interest point.

To get a better view on the performance with the interest points, we compare the computational

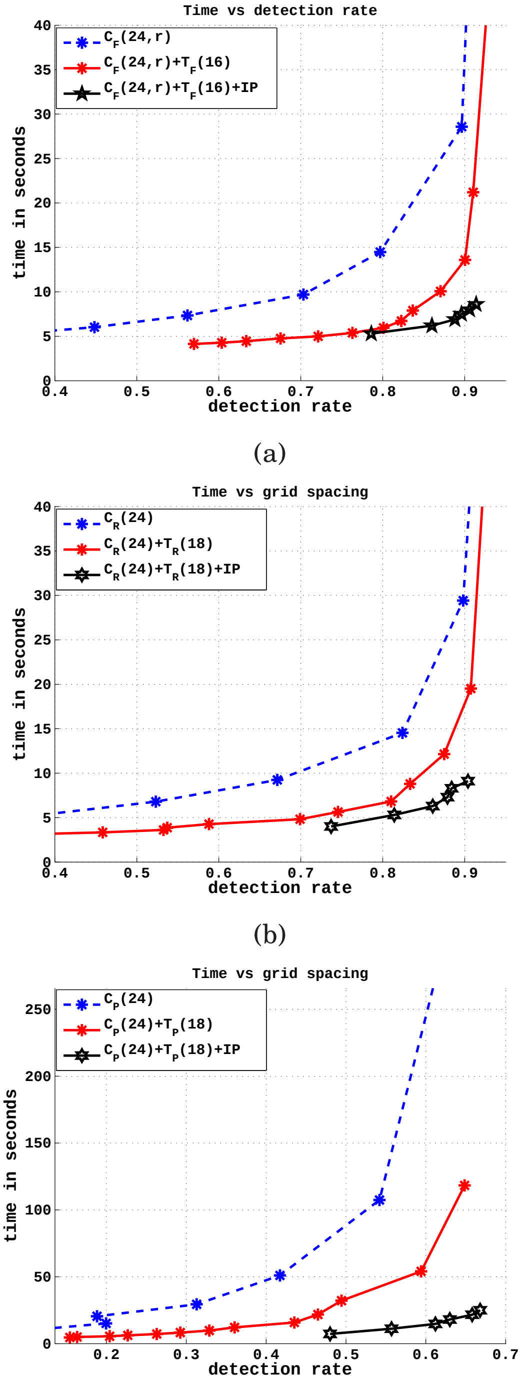
Number of sub-windows tested	detection rate	time in seconds	number of false detections
856,024 ($\mathcal{O}_{1:6}$)	90.4%	9.1	55
739,510 ($\mathcal{O}_{1:5}$)	88.4%	8.3	47
619,335 ($\mathcal{O}_{1:4}$)	87.9%	7.3	34
492,949 ($\mathcal{O}_{1:3}$)	86.1%	6.3	26
342,566 ($\mathcal{O}_{1:2}$)	81.4%	5.3	22
182,912 ($\mathcal{O}_{1:1}$)	73.1%	4.0	15

Table 5.3. Evaluation of the detection rate, speed and the number of false detections using interest point with different number of image patches on the CMU rotated face database.

Number of sub-windows tested	detection rate	time in seconds	number of false detections
1,601,024 ($\mathcal{O}_{1:6}$)	66.8%	25.0	352
1,374,136 ($\mathcal{O}_{1:5}$)	65.0%	21.6	305
1,138,398 ($\mathcal{O}_{1:4}$)	63.0%	18.6	257
875,930 ($\mathcal{O}_{1:3}$)	61.2%	14.8	221
600,943 ($\mathcal{O}_{1:2}$)	55.7%	11.2	159
310,057 ($\mathcal{O}_{1:1}$)	48.0%	7.3	110

Table 5.4. Evaluation of the detection rate, speed and the number of false detections using interest point with different number of image patches on the CMU profile face database.

time with respect to the detection rate for regular grid spacing, which are shown in Figure 5.6(a), (b) and (c) for the CMU frontal, rotated, and profile face database respectively. From these figures, we can observe that when searching for faces with interest point detection, we can achieve a better detection rate for a fixed computation time when compared to the detection rate obtained by scanning using location estimation on a regular grid. For example, in Figure 5.6(a), for a detection rate of 90%, the computation time with interest points is 1.58 times faster than scanning using location estimation on a regular grid, and 3.3 times faster than scanning without location estimation on a regular grid. A similar observation is seen in Figures 5.6(b). In the case for the CMU profile database, using interest points achieves close to baseline performance with the grid spacing of 1. The computation time for a fixed detection rate of 66.8% with interest points is around 4.8 times faster than scanning using location estimation on a regular grid, and 16 times faster than scanning without location estimation on a regular grid. Note that here we have compared with the grid spacing of 1, hence we obtain a larger gain with respect to the computational time. Some examples of face detection using interest points are shown in Figure 5.7.



(a)

(b)

(c)

Figure 5.6. Comparison of computational time vs. detection rate on the CMU face databases for different scanning approaches. (a) for the CMU frontal face database, (b) for the CMU rotated face database, and (c) for the CMU profile face database. The legend with "IP" represents scanning with interest point approach.

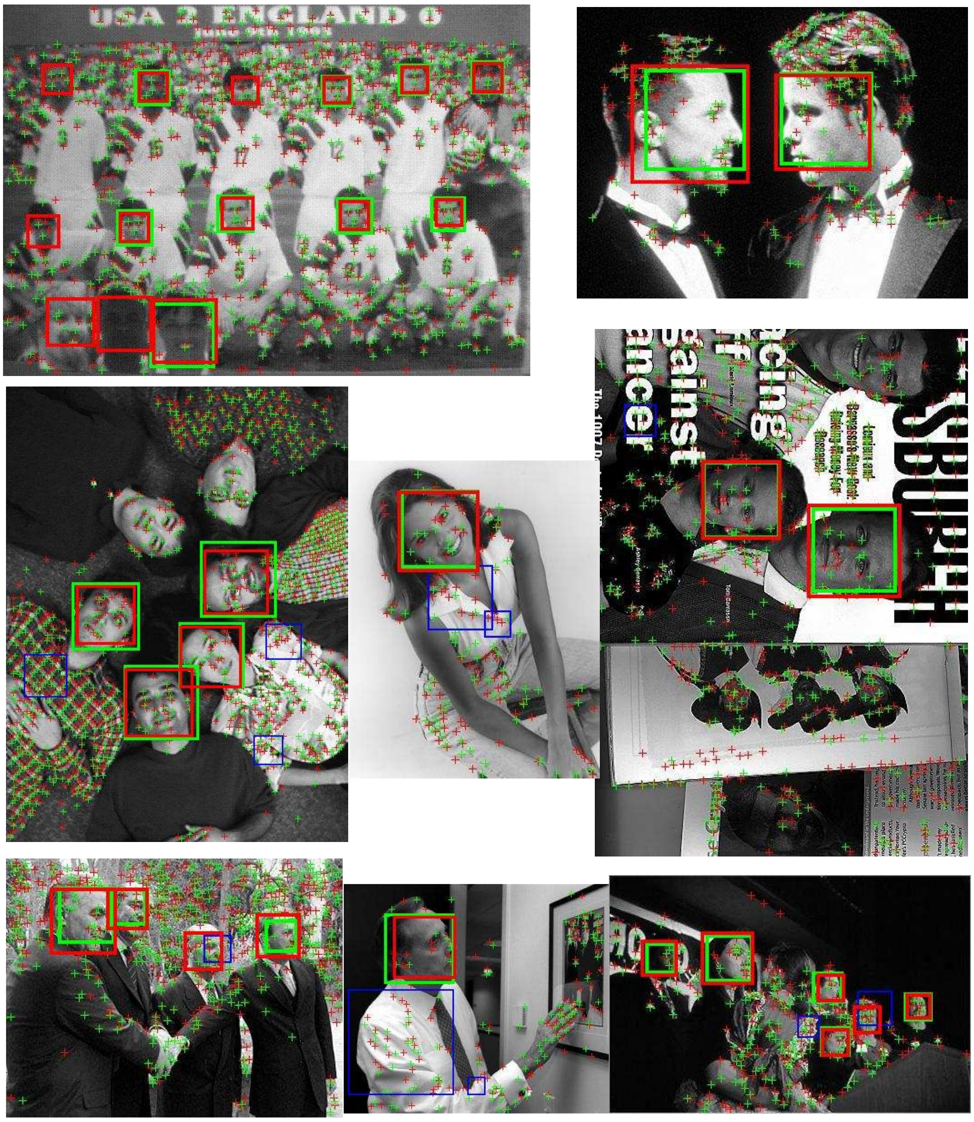
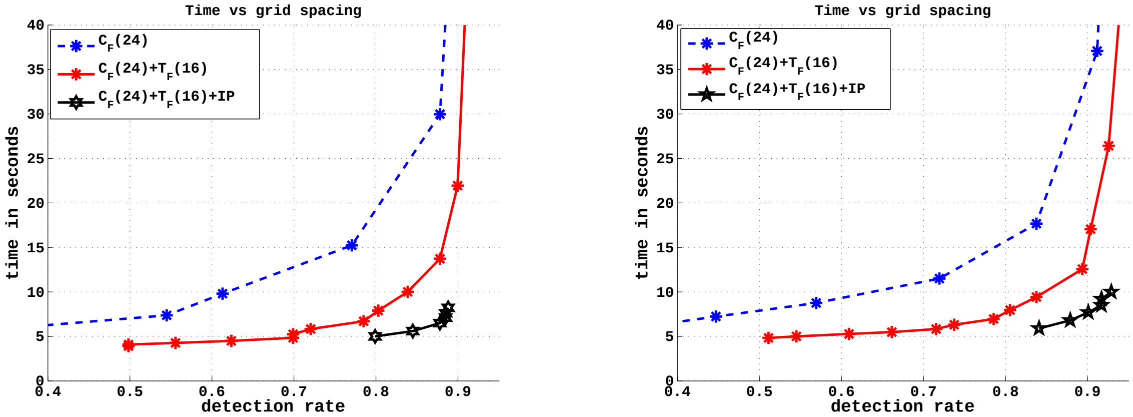


Figure 5.7. Some examples of face detection using interest points on the CMU face database. The green and red "+" marks represents the interest points detected, the red bounding boxes represent the ground-truth, the green bounding boxes represents the detected faces, and the blue boxes represents false detections.

5.4 Experiments on additional face databases

In this section, we report the results with interest point for the Cinema and Web, and CMU Multi-PIE face databases. To evaluate on these databases, the experiment setup is kept the same as described in Section 5.3.1.

Cinema and Web: Figure 5.8 shows the computational time with respect to the detection rate on the Cinema and Web frontal face databases. The results with interest point detection is compared with the results from the Section 4.4.1 (standard scanning approach with and without location estimation). From these figures, we can again observe that a good detection rate with lesser computational time is achieved with the use of interest point detection when compared to the other two approaches. For example, in Figure 5.8(b) (result on Web face database), for a fixed detection rate of 92.5%, using interest point detector speeds up the detection by 2.5 times when compared to scanning with location estimation, and it is 3.75 times faster than the standard scanning technique with out any location estimation. Some detection results for this database along with interest point overlaid are shown in Figure 5.9.



(a) (b)

Figure 5.8. Comparison of computation time vs. detection rate on the Cinema and Web frontal face databases for different scanning approaches. (a) for the Cinema frontal face database, (b) for the Web frontal face database. The legend +IP represents scanning with interest point approach.

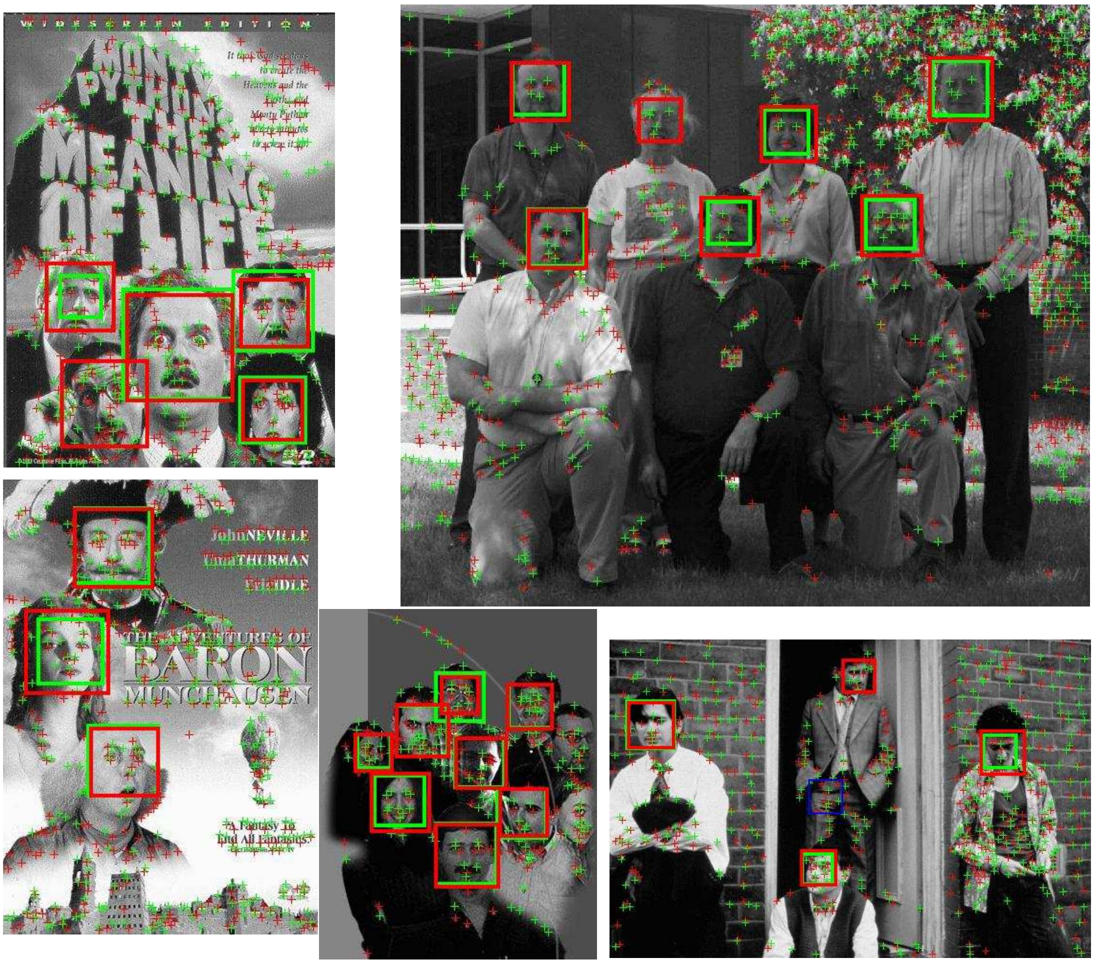

WIDESCREEN EDITION
MONTY PYTHON THE MEANING OF LIFE
John NEVILLE
THURMAN
THE ADVENTURES OF BARON MUNCHAUSEN
A Fantasy To End All Fantasies

Figure 5.9. Some examples of face detection using interest points on the Cinema and Web face database. The green and red "+" marks represents the interest points detected, the red bounding boxes represent the ground-truth, the green bounding boxes represents the detected faces, and the blue boxes represents false detections.

CMU Multi-PIE The computational time with respect to the detection rate for Multi-PIE database are shown in Figure 5.11. We can see from the figure that, when using interest point detector to detect profile faces there is a drop in detection rate compared to the scanning with grid spacing of 1. This might be due to the lesser number of image samples that we considered around and interest point for profile views. In the case of frontal face detection, the performance is close to the baseline with much lesser computational time. Some detection results with interest point overlaid are shown in Figure 5.10.

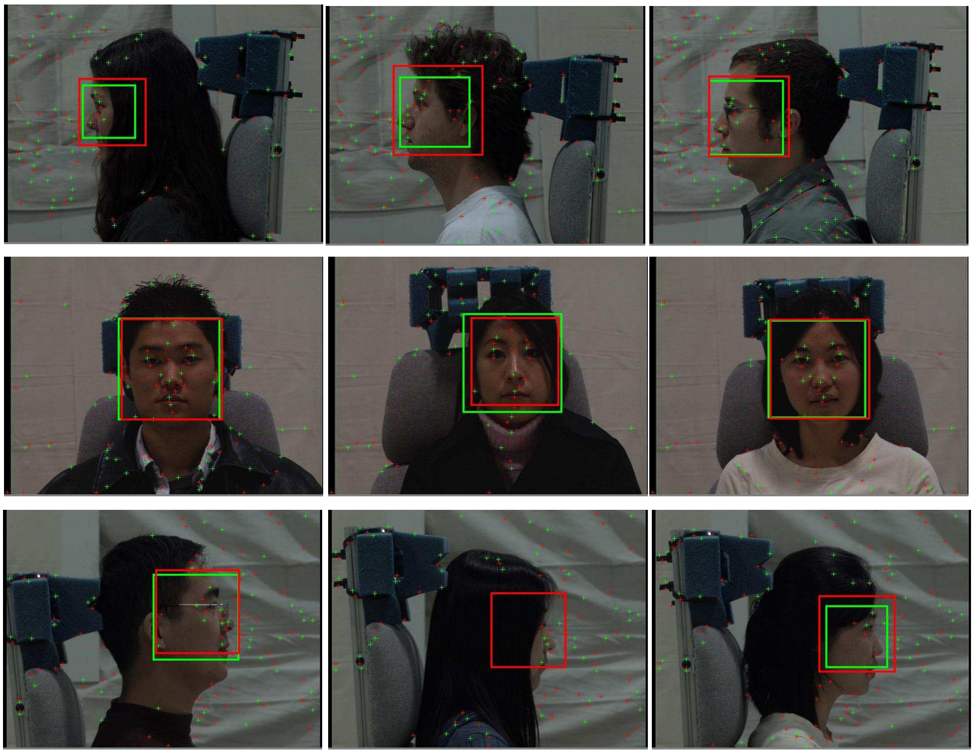
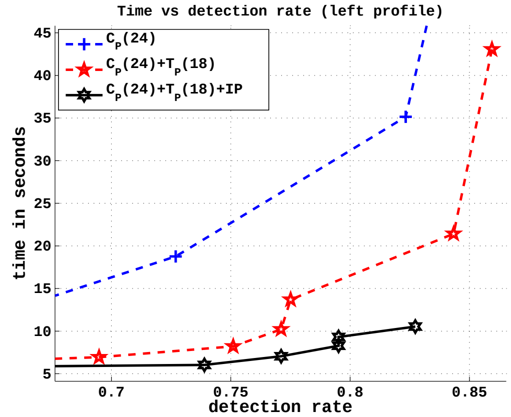
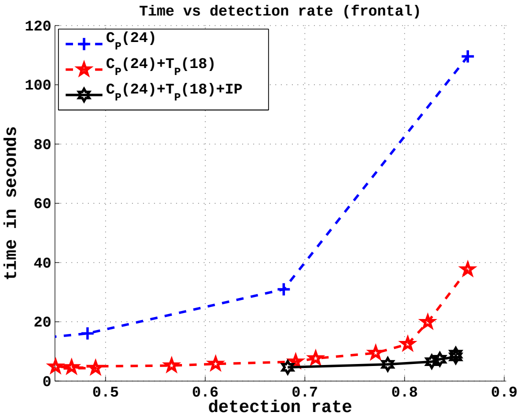


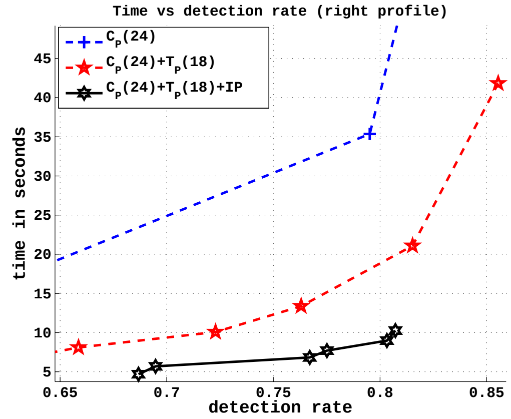
Figure 5.10. Some examples of face detection using interest points on the Multi-PIE face database. The green and red "+" marks represents the interest points detected, the red bounding boxes represent the ground-truth, the green bounding boxes represents the detected faces, and the blue boxes represents false detections.



(a)



(b)



(c)

Figure 5.11. Comparison of computation time vs. detection rate on the the Multi-PIE face database for different scanning approaches. (a) for the left face profile, (b) for the frontal face, and (c) for the left face profile. The legend "IP" represents scanning with interest point approach.

5.5 Summary

In this chapter we proposed a method on improving the search technique using location estimation and interest points. We proposed a simple interest point detector based on quantized gradient orientation. A few image patches are sampled around an interest point which is used by the location estimator to predict the face location which is further verified by a strong classifier. Interest points detected can be looked as a non-regular grid compared to the regular grid as in the sliding windows technique. We have shown that using interest point and location estimation can enhance the detection rate and speed on various benchmark face databases.

Chapter 6

Conclusion and future work

The standard approach of detecting faces from an image is based on the sliding window technique. To search for faces from an image, a classifier is evaluated at every location and scale. Cascade of classifiers are usually used to speed up the detection process. The other approach to speed up the detection process is to reduce the number of subwindows evaluated. In the sliding window technique, the number of subwindows processed can be controlled by varying the grid spacing, but as the grid spacing is increased (fewer subwindows processed) the number of miss detections also increases.

In this thesis, we proposed an alternative search technique using location estimation to reduce the number of miss detections when fewer number of subwindows are processed in the sliding window framework. A decision tree was used to estimate the location of the face by analysing an image patch. We also investigated the use of simple interest point detector based on quantized gradient orientation to improve the proposed search technique.

6.1 Location estimation using a decision tree and binary features

A decision tree was used to learn the association between the patch appearance (a part of a face) and its location (offset) within the face image. We carried out several experiments with different patch sizes, depths of the tree and for two binary features (ferns and μ-ferns). We evaluated its performance on frontal, in-plane rotated, and out-of-plane rotated cropped face database. Additionally, a decision tree was also used to estimate the view of the face.

Location estimation: In the case of frontal faces, location estimation obtains a good performance with a smaller depth of the tree, but when considering multiple views (frontal in-plane and out-of-plane rotated faces) the tree depth has to be larger to learn the variations in the face appearance across different views for the same offset value. For smaller patch sizes the performance of the location estimation drops within a specified offset error, since the number of offset parameters increases and also a smaller patch size might become less discriminative from each other, when compared to the larger patch size.

Binary feature: In all the experiments, the proposed binary feature μ-ferns performs comparable to the fern feature. The computation time for the decision trees with the two binary tests were evaluated empirically and the results revealed that a decision tree with μ-ferns was slightly faster when compared to a decision tree with fern feature, for a fixed depth of the tree.

Face view estimation: We used the same binary feature and learning process for location estimation to learn the association between the patch appearance and the face pose (in-plane and out-of plane rotation). A decision tree was learnt to estimate 19 views for in-plane rotation, and 3 views for out-of-plane rotation. On the cropped face database the performance of the face pose estimation was also reasonably good within a small error in estimation.

6.2 Application to face detection

We applied our proposed search technique to detect faces from an image. For this purpose, we used a cascade of boosted multi-block LBP features as a strong view-specific classifier. The location estimation with different patch sizes along with a baseline classifier were evaluated on different face databases.

Frontal face: The effect of the location estimation on the detection rate for different grid spacing was more visible for frontal face detection compared to the rotated and profile face detection. In the case of face detection on frontal face databases, scanning at very large grid spacing could also achieve very good performance. The effect of the patch sizes for larger grid spacing was seen on various frontal face databases. It is interesting to note that, though smaller patch sizes have less accuracy in estimating the location of the face, when used in the scanning framework with larger grid spacing, the detection rate achieved is better than using larger patches. However, beyond a certain patch size, the difference in performance is not clearly visible.

Frontal in-plane rotated faces: The baseline classifier in this case consists of an in-plane rotation estimator and 19 view-specific classifiers. The performance of the baseline classifier was good with reasonable number of false positives, which also shows that the in-plane rotation estimator was performing well on this face database. The use of location estimation improved the detection rate for larger grid spacing, but not as much as obtained for frontal faces. Also, the effect of patch sizes on the detection rate for larger grid spacing is not very clear on the CMU rotated face database.

Profile faces: The baseline classifier in this case consists of an out-of-plane rotation estimator and 3 view-specific classifiers. The overall performance of the baseline classifier was not that good when compared to the frontal and in-plane rotated baseline classifiers. The effect of the location estimation on the detection rate for different grid spacing was the lowest on the CMU profile face database. The variability in the appearance for profile views is much more than frontal and frontal in-plane rotated faces, which makes learning a good baseline classifier and also a location estimator for detecting profile faces a challenging task.

We also evaluated the out-of-plane face detector along with location estimation on a subset of the CMU Multi-PIE database, which contains equal number of left and right profile faces. On this database the performance of the baseline classifier was reasonably good. The effect of the location estimation on the detection rate for the profile views was also clearly seen on this database.

On improving the proposed search technique: Instead of using the regular grid spacing, we also proposed the use of interest point to improve the proposed search technique. The interest point detector is based on quantized gradient orientation which is fast to compute. Experiment results on various face databases showed good performance with respect to the detection rate and speed when compared to the regular grid spacing in the sliding window framework with and without location estimation.

In conclusion, in all the various face databases, the proposed approach could reduce the number of miss detections for a larger grid spacing compared to the standard search technique. We also could gain in speed for a fixed detection rate since the location estimation was faster than the baseline classifier. We also observed that scanning with the grid spacing of 1 still achieves the best detection rate for the baseline compared to the proposed approaches, but for a small drop in performance, our approach can achieve a better computational time, which might be useful for practical purposes.

6.3 Possible directions for future work

Possible directions for future work are outlined below:

Mobile application: The proposed search technique, within the sliding window framework along with location estimation could be tested on mobile devices where the computation resource is limited. The application of the proposed or other faster interest point detector for face detection (in-plane and out-of-plane rotated faces) could also be evaluated on a mobile device.

Using an improved strong classifier for face detection with location estimation: From our experiments detecting profile faces seems to be a challenging task. We have only used multi-block LBP features for this task and a different set of features needs to be explored to improve the detection performance.

Using alternative features for location and pose estimation: We have currently used only simple binary features for estimating the location and the pose of the face. Other features (such as the gradient orientation) could also be explored to see if there are any improvement in performance.

Investigating interest point detector for face detection: We evaluated the face detection performance using a simple interest point detector based on the gradient orientation information. Quantizing gradient orientation in different angles could be further explored to detect interesting feature location on the face region which could be then used for the task of face detection. Also, state-of-art interest point detectors along with location estimation could be evaluated for face detection task. As we have seen that using interest point achieves better performance (speed and accuracy), this approach could be evaluated on high resolution images and videos.

Face detection in general: Most of the current methods to train a face detector require annotated face databases. Approaches based on unsupervised methods for clustering would be more appropriate to build a complete multi-view face detection system.

Appendix A

Feature Extraction

Features are used to describe patterns in a more compact way, such that the intra-class variability of the patterns is decreased and inter-class variability is increased. Features extracted for object detection should be robust to illumination changes, and ideally require as little computation as possible. In pioneering work, gray scale pixel values were used directly by Rowley [84], Feraud [78], and Osuna [67] and complemented by an image preprocessing (histogram equalization, smoothing). This feature extraction mechanism was computationally very expensive. In this following sections, we will discuss on different features which are being used for object detection and tracking. We will not be able to cover all types of features, but select a subset which are simple and efficient to compute. We will discuss about Haar-like features, Local Binary Patterns (LBP) and its variants, and ferns.

A.1 Haar-like features

The first real-time frontal face detection introduced by Viola and Jones [107] used haar-like feature for face detection. Haar-like features (ref. Figure A.1(a)) are derived from Haar wavelets and a feature is computed by subtracting the sum of pixel values in the white and the black region. Those features can be efficiently computed by using an *integral image* or *summed area table*, first introduced by Crow [15] for texture mapping. At a given location (*x,y*), in an image, the value of the *integral image ii*(*x,y*) is the sum of the pixels above and to the left of (*x,y*):

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x',y')$$

where $i(x',y')$ is the pixel value of the original image at location (x',y'). To compute the sum of the black region shown in Figure A.2, only 4 table access and 3 simple operations are needed. Following Viola and Jones, Lienhart et al. [55] introduced rotated haar-like features (Fig. A.1 (b)) which significantly enriched the original haar-like feature and could be calculated also very efficiently.

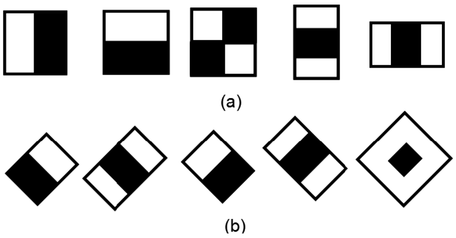


Figure A.1. (a) Haar-like features used by Viola and Jones, and (b) rotated haar-like features used by Lienhart in addition to original haar.

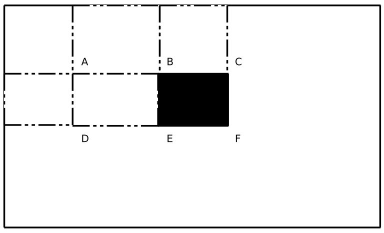


Figure A.2. Computation of Haar-like feature with the *integral image*. The sum in the black region is given by: $F - C - E + B$

A.2 Local binary patterns

Another type of feature that are popular in the computer vision community are for instance Local Binary Pattern (LBP) [66], Census Transform [77], and Modified Census Transform (MCT) [25]. These features are non-parametric operators which captures the local spatial structure around a pixel in an image. For example, the LBP operator is represented by

$$LBP_{P,R}(x_c,y_c) = \sum_{p=0}^{P-1} s(g_p - g_c)2^p$$

where g_c corresponds to the gray value of the center pixel (x_c, y_c) of a local neighbourhood, $g_p(p = 0, ..., P-1)$ corresponds to the gray values of P equally spaced pixels on a circle of radius R $(R > 0)$ that forms a circularly symmetric set of neighbours. The function s is defined as:

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

Figure A.3 shows the computation of $LBP_{8,1}$. A modification of $LBP_{8,1}$ was introduced by Froba et al. [25] called Modified Census Transform (MCT), which can be written as

$$MCT(x_c, y_c) = \sum_{p=0}^{8} s(g_p - g_p^*)2^p$$

where g_p consists of center pixel and its 8 neighbours, and g_p^* is the average of all the pixel values in g_p. MCT thus has an extra bit to encode the local structure. The number of binary comparisons for $LBP_{8,1}$, $LBP_{4,1}$, and MCT are 8, 4, and 9 respectively. Figure A.4 shows the local primitives detected by $LBP_{8,1}$ operator (spots, line end, edges, and corner). In terms of texture, each LBP code can be regarded as *micro-textons*. These binary features have shown robustness to monotonic gray scale transforms. The representation of LBP extended to multi-scale is known as multi-block LBP feature, where instead of a pixel value the sum of a block is considered. The sum over block can be efficiently computed using integral images.

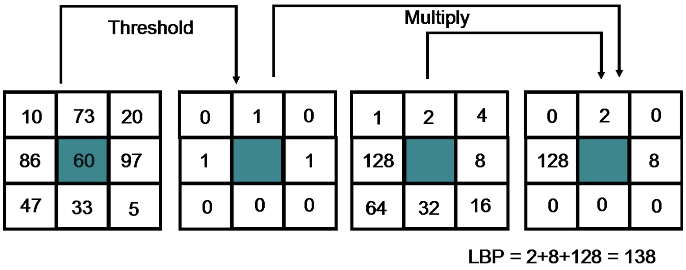


Figure A.3. Computation of $LBP_{8,1}$. A LBP code for a neighborhood is produced by multiplying the threshold values with weights given to corresponding pixels and summing up the result.

A.3 Ferns

Özuysal et al. [68] have shown that simple binary features could be used for patch classification.

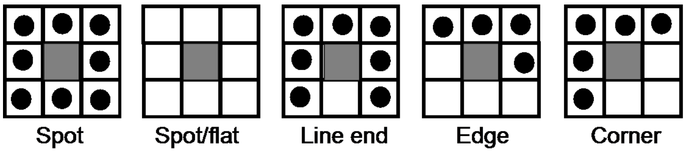


Figure A.4. Different texture primitives detected by LBP. In the figure ones are represented by white square and zeros by black circles.

The binary features they proposed are based simply on the sign of pixel intensity difference. Given an image patch I a binary feature is defined as

$$f_i = \begin{cases} 1 & \text{if } I(m) \leq I(n) \\ 0 & \text{otherwise} \end{cases}$$

where $I(m)$ and $I(m)$ are the pixel intensity values at location m and n. A *fern* is defined as a set of binary features (see Figure A.5), and can basically capture the intensity variations in an image patch. A set of location pairs (m, n) can be structured vertically, horizontally, diagonally, or circularly in a similar manner to LBP or to any arbitrary shape.

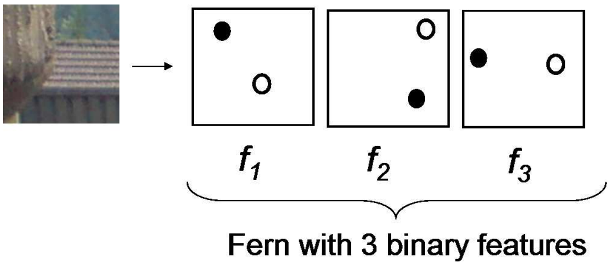


Figure A.5. Fern feature. The dark and light circles shown in a white boxes are arbitrary pixel location (m and n). In this figure the fern consists of 3 binary feature (f_1, f_2, and f_3).

Appendix B

Face detection using ferns

In this work, we propose to use fern feature for face detection task. This feature is motivated by [68], in which a fast patch classification algorithm based on Semi-Naive Bayesian classifier and binary features (ferns) was proposed for estimating the pose of an object. Our main motivations to use this feature was that it does not require any preprocessing of the image and the training time is greatly reduced when compared to boosting.

In [68], the pixel pair location for computing the binary feature were selected randomly. However, random selection of binary features does not help us to achieve consistent performance for face detection task and it will not be optimal for building a cascade of classifier. In next section, we first describe the binary feature selection process, then describe Semi-Naive Bayesian classifier using fern features, and finally describe the experiments conducted for face/non-face classification.

B.1 Binary feature selection using conditional mutual information

For a image of size $q \times q$ we have $q(q-1)/2$ possible binary features. The main goal of feature selection is to select a small subset of features that carries as much information as possible. The ultimate goal would be to choose $f_1, ..., f_{N_f}$ which minimizes $H(Y|f_1, ...f_{N_f})$, where H is the entropy, Y the class label (0,1), f_i binary feature (0,1), and N_f number of binary features. But this expression can not be estimated with a training set of realistic size as it requires 2^{N_f+1} probabilities (considering all $N_f + 1$ (total number of features plus the class) binary combination).

Fleuret et al. [24] showed that features selected based on conditional mutual information maximization (CMIM) criteria and using a Naive Bayesian classifier gives performance comparable to state-of-art techniques such as boosting or SVM. The features selected using CMIM also show robustness to noisy training data. Conditional mutual information is given by

$$I(Y; f'|f) = H(Y|f) - H(Y|f', f) \tag{B.1}$$

where $H(Y|f)$ is the conditional entropy which measures the residual uncertainty of Y when f is known, $H(Y|f', f)$ measures the residual uncertainty of Y when f and f' are known, and $I(Y; f'|f)$ gives an estimate on the amount of information shared between Y and f' when f is known. The conditional mutual information is zero if f' and f have the same information about Y, while the value is large if f' and f have different information about Y. To select a feature that carries different information from features that are already selected, the following iterative scheme was proposed in [24],

$$\begin{aligned} f_0 &= \arg\max_n I(Y; f_n) \\ \forall k, 0 \le k \le K-1, f(k+1) &= \arg\max_n \underbrace{\{min_{l \le k} I(Y; f_n|f_l)\}}_{s(n,k)} \end{aligned}$$

The score $s(n, k)$ is low if any feature already picked is similar to f_n or if f_n does not contain any information about Y. By taking the feature f_n with maximum score $s(n, k)$, it is ensured that the new feature is different from those that are already selected and also carries information about Y.

B.2 Semi-Naive Bayesian classifier

Naive Bayesian classifier has been successfully used for various classification tasks. We will be using Semi-Naive Bayesian technique proposed in [68] to build a two class classifier. The idea is to build the class conditional probabilities of binary feature and at run-time use these probabilities to select a pattern with highest likelihood. We will first define the following notations:

f_i : Binary feature (sign of intensity difference of two pixels).

c_i : Class label

F_j : A Fern defined to be a set of S binary features $\{f_l, ..., f_{l+S}\}$

M : Number of ferns

$N = S \times M$, where N is total number of features in the model.

Given a set of features $f_0, f_1, ..., f_{N-1}$ the idea is to select class c_i such that

$$\hat{c_i} = \arg\max_{c_i} P(C = c_i | f_0, f_1, ..., f_{N-1}) \tag{B.2}$$

Bayes' Formula yields

$$P(C = c_i | f_1, f_2, ..., f_N) = \frac{P(f_0, f_1, ..., f_{N-1} | C_k) P(C_k)}{P(f_0, f_1, ..., f_{N-1})}$$

Assuming a uniform prior $P(C)$ and since the denominator is simply a scaling factor that is independent from the class the problem reduces to finding

$$\hat{c_i} = \arg\max_{c_i} P(f_0, f_1, ..., f_{N-1} | C = c_i) \tag{B.3}$$

The joint probability of EquationB.3 is not feasible since it would require estimating and storing 2^N entries for each class. One way to simplify the representation is to assume independence between features.

$$P(f_0, f_1, ..., f_{N-1} | C = c_i) = \Pi_j^{N-1} P(f_j | C = c_i).$$

However, this completely ignores the correlation between features. To make the problem tractable while accounting for these dependencies, a compromise is to partition the features into M groups of size $S = \frac{N}{M}$. These groups are defined as *ferns* and the joint probability for feature in each fern is computed. The conditional probability becomes

$$P(f_0, f_1, ..., f_{N-1} | C = c_i) = \Pi_k^{M-1} P(F_k | C = c_i).$$

where $F_k = f_{k,0}, f_{k,1}, ..., f_{k,S-1}, k = 0, ...M - 1$. The parameters M and S could be used to tune the performance and memory trade-off.

B.3 Experiments

B.3.1 Database

The database consists of 8744 training and 9232 validation faces, taken from different face database (BANCA [6], Essex, Feret [73], ORL, Stirling and Yale [8]). The face samples are aligned with respect to eye coordinates. For non-face patterns we randomly select around 50000 patterns from different images not containing any face. All the patterns are resized to $q \times q$ pixels, where $q = 19$. For testing the performance of the classifier we have three different test sets. Test set 1 contains 2360 faces with frontal illumination taken from XM2VTS database [62]. Test set 2 contains 1180 faces with side illumination (XM2VTS darkened). Test set 3 contains 580 faces from various sources (web). For non-face test patterns we took MITCBCL[1] dataset and combined both training and test non-face set to obtain 27000 patterns. Samples from these dataset are shown in Figure B.1.

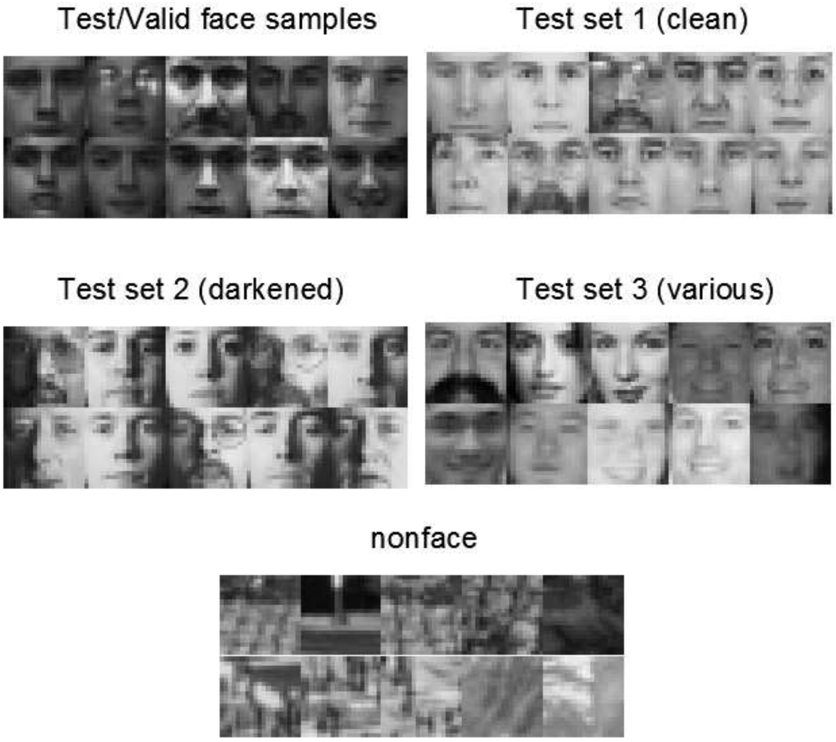


Figure B.1. Face and non-face samples form training, validation and test dataset.

---

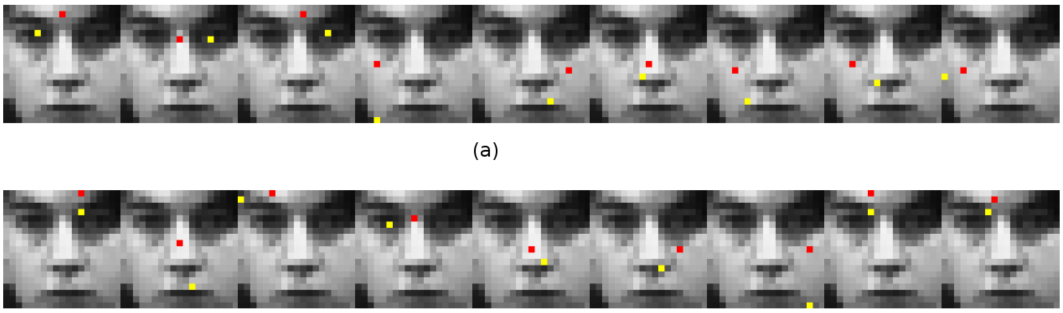
[1] MIT Center For Biological and Computation Learning, http://www.ai.mit.edu/projects/cbcl

B.3.2 Training and testing

Face detection is a two class problem, therefore we have $C = \{0, 1\}$. From the training face and non-face patterns, we select N features, $N << 64890$ (for $q = 19$ we have 64890 possible binary features) using CMIM criteria as described in Section B.1. There are many ways in which the N features can be grouped in size of S. The N features are split into M equal sets, each of size S to form each fern F_j, $F_j = f_{h+0}, ..., f_{h+S-1}$, where $h = j \times S$, $j = 0, ..., M-1$. An example of pixel pair locations of first two ferns ($S = 9$) obtained after selection using CMIM are shown in Figure B.2. Once the selection is done, it becomes possible to build the class conditional probability $P(F_j|C = c_i)$ for all ferns using the training patterns of face and non-face. To test if a pattern is a face or a non-face we check the following condition:

$$\frac{\Pi_k^M P(F_k|C = 1)}{\Pi_k^M P(F_k|C = 0)} > \tau \tag{B.4}$$

The value τ is found using the validation set for a given detection rate. The detection rate or true positive rate (TPR) is defined as the percentage of faces that are correctly accepted in a particular data set. False positive rate (FPR), also called false alarm rate is defined as the percentage of non-face patterns accepted as face. Classification error rate is defined as the percentage of total number of miss classification (face classified as non-face and non-face classified as face).



(a)

(b)

Figure B.2. Illustration of fern features overlapped on face image. (a) shows the pixel pair location for first fern, and (b) shows pixel pair location for second fern. Both the ferns shown here have $S = 9$.

B.3.3 Results for single stage classifier

To compare the performance of ferns and other features (MCT, haar-like feature, and LBP) we plot the receiver operating characteristic (ROC) curves and look at the TPR and FPR on the test dataset. MCT, LBP and haar-like features are trained using boosting techniques. To have a fair comparison among different techniques we selected equivalent number of features in each case. For our approach, we tried different values of S and M and selected $S = 9$ and $M = 10$, which performed better (see Figure B.3). We have a total of 90 binary features for $S = 9$ and $M = 10$. Therefore, we set all other techniques to have equivalent to 90 binary features: the number of features for MCT is 10, 11 for $LBP_{8,1}$, and 22 for $LBP_{4,1}$. In the case of haar-like features it was difficult to decide on the exact number of features due to different ways the features were computed. We decided to take approximately 50 haar-like features based on computation cost between the binary features, and 100 features to see roughly the performance when more features are added.

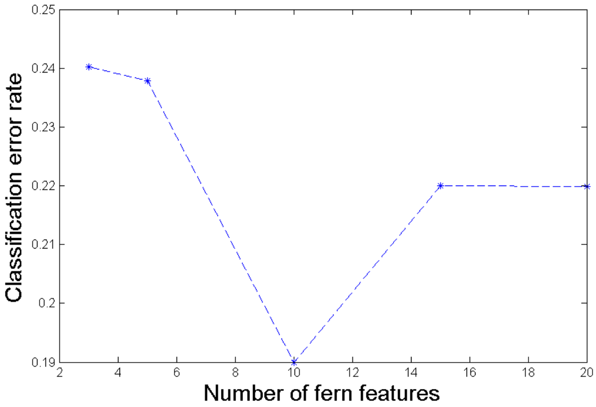


Figure B.3. Classification error rate vs. number of fern features (3,5,10,15, and 20) for $S = 9$

The performance between MCT, ferns, $LBP_{8,1}$, $LBP_{4,1}$, and haar-like features are shown in Figures B.4, B.5, and B.6, and Table B.1. To obtain TPR and FPR in Tab. B.1, the decision threshold for each classifier was obtained by setting the detection rate to 98% on the validation dataset. From the ROC curves and the Table B.1, we see that haar-like features when compared to binary features are not so robust to changes in illumination. The performance of ferns and MCT features are comparable across different test data set.

Table B.1. True and False positive rate for Test sets in %. The first column shows the feature type, column second, third and fourth show the corresponding TPR for test set 1, test set 2, and test set 3 respectively, and finally the last column shows FPR obtained on non-face test set.

	Test set 1	Test set 2	Test set 3	FPR
MCT	98.22	96.78	95.69	12.25
ferns	99.53	99.15	96.9	16.9
$LBP_{8,1}$	97.88	97.03	96.55	22.80
$LBP_{4,1}$	99.32	95.76	95.34	20.12
haar 100	99.41	89.14	94.83	8.53
haar 50	99.58	86.65	93.28	21.01

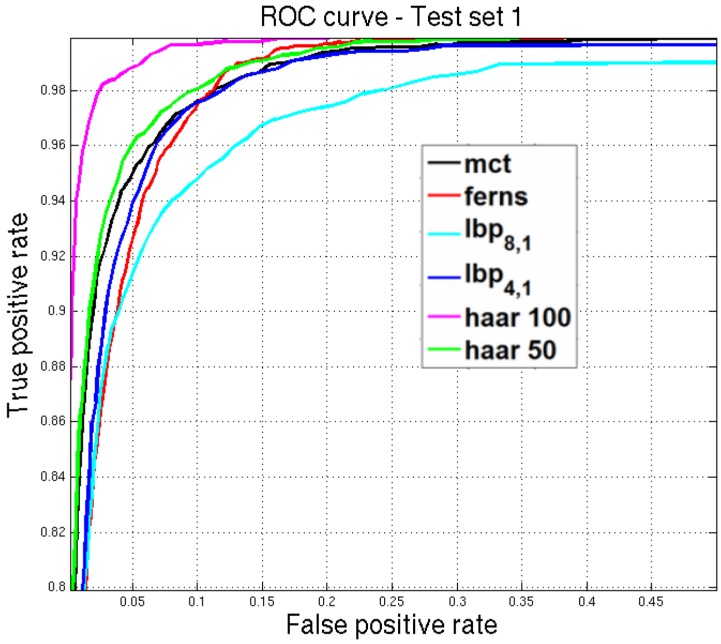


Figure B.4. ROC curve on the XM2VTS (normal) database (Test set 1).

B.3.4 Cascade model architecture and results

Finally, we built a cascade of classifiers to reduce the false positive rate while maintaining a high true positive rate. The hyper-parameters of the cascade were selected empirically. A 40 stage cascade with a total of 800 fern feature ($S = 9$) was necessary to obtain a reasonable false positive rate of 10^{-4} by keeping the detection rate at each stage to 99.91% on training data set. The same face training and validation dataset as described in section B.3.1 were used, while non-face samples were obtained by bootstrapping at each stage from a set of 1734 images containing no faces. We look at the performance of the cascade model on cropped faces and also by scanning full images

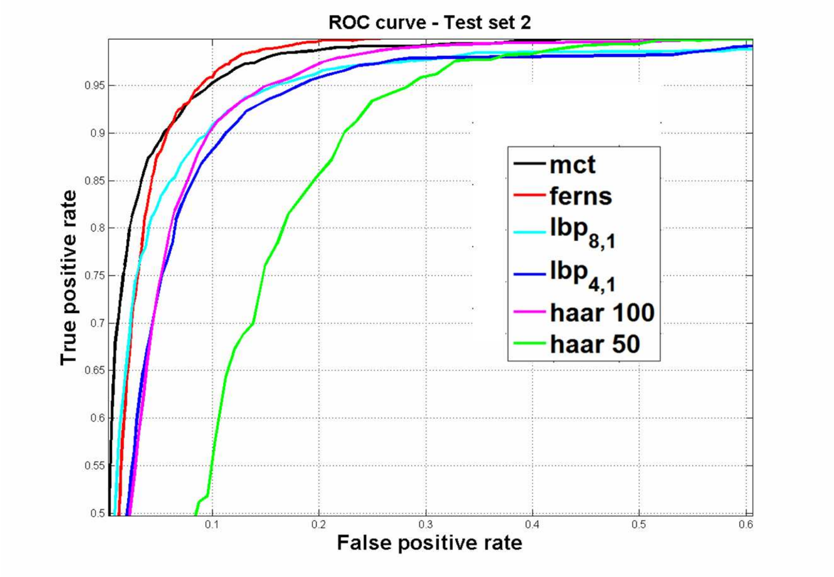


Figure B.5. ROC curve on the XM2VTS (darkened) database (Test set 2).

from the BANCA (English corpus) and the XM2VTS databases. We used sliding window approach at different scales (pyramid scan) to detect faces from an image. The accuracy of localization on the detection result is given by measuring the difference between the ground truth eye center location with the estimated one [41]. The localization accuracy measure is given by

$$d_{eye} = \frac{\max(d(C_l, C_l^*), d(C_r, C_r^*))}{d(C_l, C_r)} \tag{B.5}$$

where $d(a, b)$ is the Euclidean distance between points a and b. C_l and C_r are the true left and right eye centers, and C_l^* and C_r^* are the estimated left and right eye centers.

Table B.2. Detection rate for various test dataset in % (rejection rate = 98.4% on non-face test set). (Note we had only cropped faces for Test set 3)

	cropped faces	localization result $d_{eye} < 0.25$ (full image scan)
Test set 1	99.06	94.9
Test set 2	98.13	85.0
Test set 3	92.07	NA
BANCA Controlled	91.1	77.9
BANCA Degraded	86.77	35.24
BANCA Adverse	95.3	86.97

The results of cascade detector are shown in Table B.2. As it can be seen from the table that, for BANCA degraded, there is a degradation in performance when the detector is applied on full

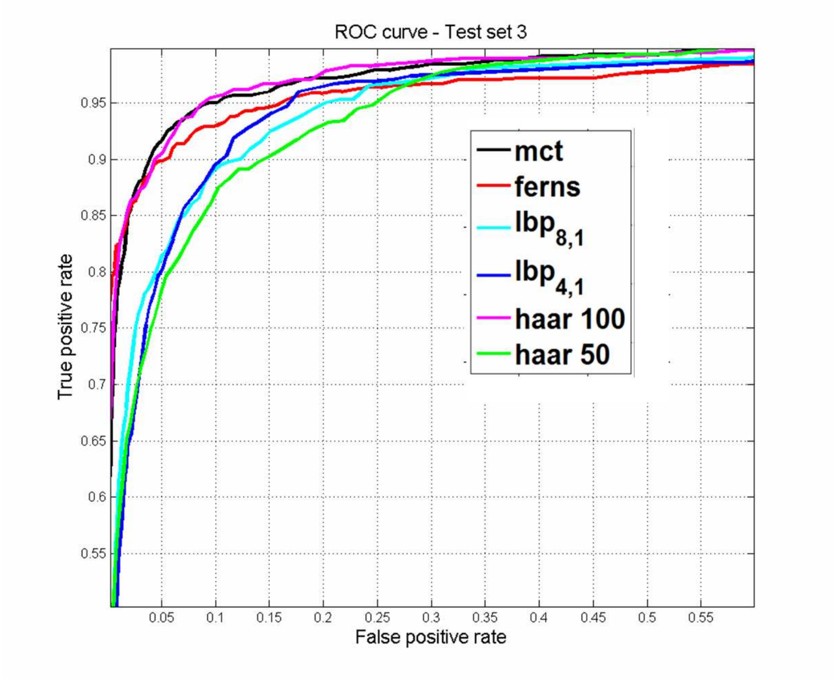


Figure B.6. ROC curve on the database collected from the web (Test set 3).

images. Some detection results are shown in Figure B.7.

In conclusion, though it is possible to build a cascade architecture using this approach for face detection, the number of stages required in a cascade is quite large, when compared to building a face classifier with boosted MCT features.

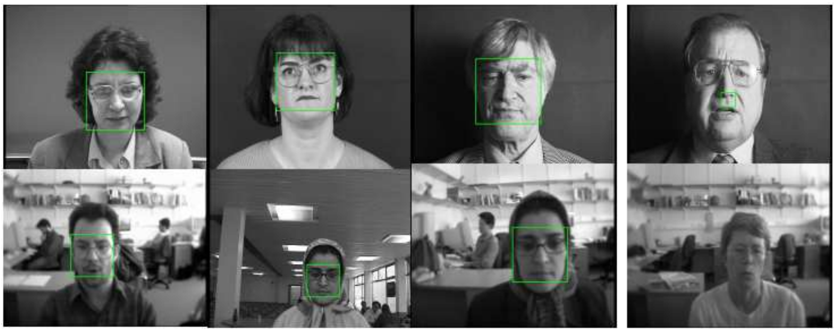


Figure B.7. Some examples of the face detection results on the XM2VTS and BANCA databases.

Appendix C

Additional results on the FDDB face database

We evaluate our approach on using interest point detector along with location estimation on the FDDB face database (ref. Section 4.4.2) with a baseline profile face classifier $C_P(24)$ (ref. Section 4.3.6 for the classifier details). The scanning and merging parameters are kept the same as described in Section 5.3.1, and 4.2.1, except for the parameter β (number of overlapping rectangles for a valid detection) which is set to 4 to reduce the number of false positives.

The FDDB database is split into 10 sets and the average ROC curve is reported in Figure C.1. We also compare with the other two detectors (Olaworks face detector and Li face detector (Intel)[52]) for which we obtained the ROC curve from the FDDB website [1]. We can see from the figure that our detector performs comparably well with respect to the other two detectors. Some detection results are shown in Figure C.2. Red boxes on the image represents the ground-truth. Note the ground-truth boxes in images 1 and 5, where the blurred faces are also annotated, which might be hard to detect. Also, the amount of occlusion in image 8 is quite high for one of the face to be reliably detected.

---

[1] *http://vis-www.cs.umass.edu/fddb/results.html*

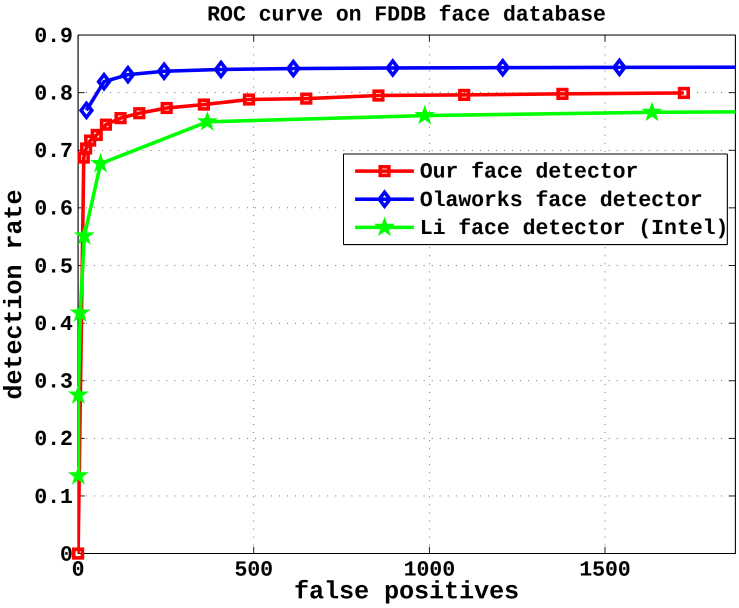


Figure C.1. ROC curve on the FDDB face database.

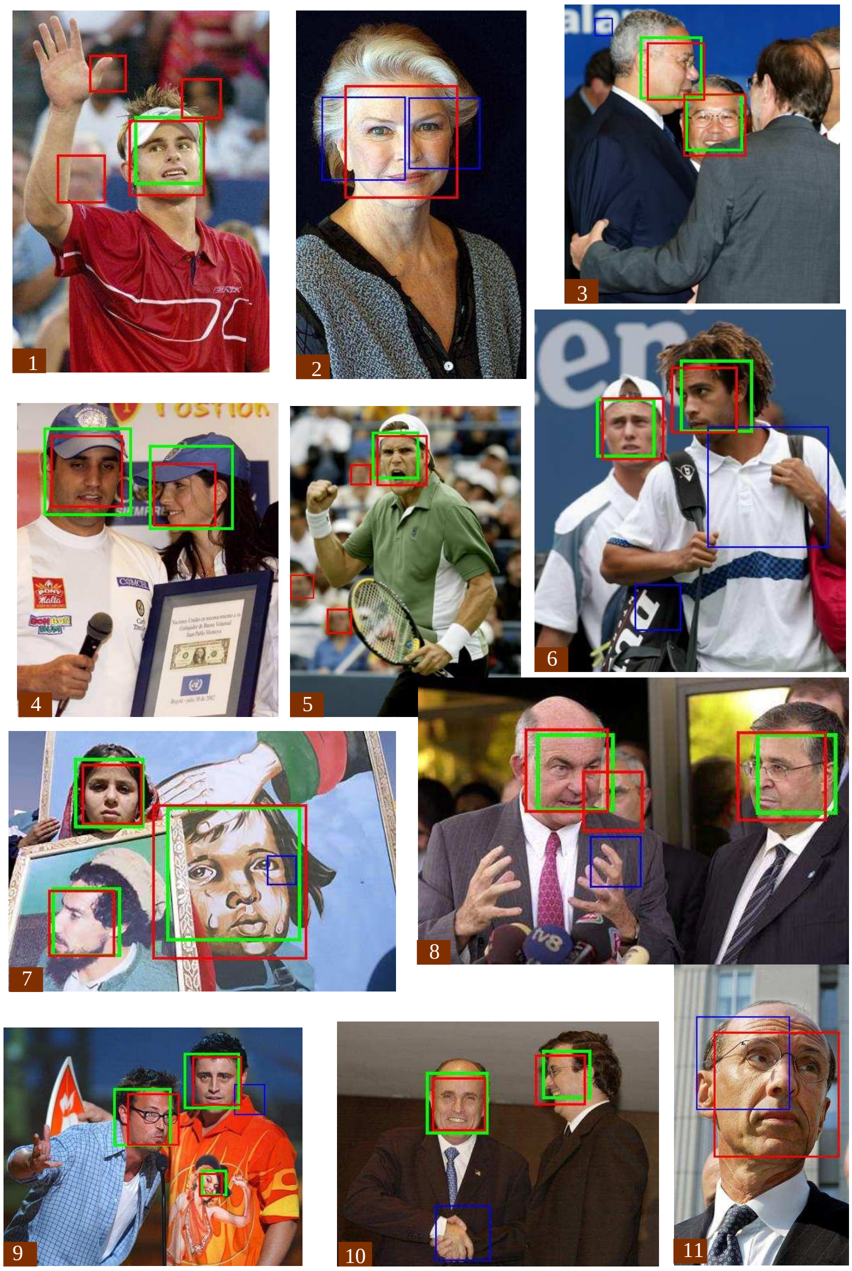


Figure C.2. Some detection results on the FDDB face database. The red boxes represents the ground-truth face location, the green boxes are the detected faces, and the blue boxes represents false detections.

Bibliography

[1] Alain, L., Bastian, L., and Luc, V. G. (2011). Fast prism: Branch and bound hough transform for object class detection. *International Journal of Computer Vision*, **94**(2), 175–197.

[2] Amit, Y. and Geman, D. (1999). A computational model for visual selection. *Neural Computation*, **11**(7), 1691–1715.

[3] Amit, Y., Geman, D., and Jedynak, B. (1998). Efficient focusing and face detection. In *Face Recognition: From Theory tu Applications*, pages 143–158. Springer-Verlag.

[4] Antonio, C., Jamie, S., Duncan, R., and Ender, K. (2011). Regression forests for efficient anatomy detection and localization in ct studies. In *Medical Computer Vision. Recognition Techniques and Applications in Medical Imaging*, Lecture Notes in Computer Science, pages 106–117.

[5] Babenko, B., Dollr, P., Tu, Z., and Belongie, S. (2008). Simultaneous learning and alignment: Multi-instance and multi-pose learning. In *ECCV Workshop: Faces in Real-Life Images*.

[6] Bailly, E., Bengio, S., Bimbot, F., Hamouz, M., Kittler, J., Mariethoz, J., Matas, J., Messer, K., Popovici, V., Poree, F., Ruiz, B., , and Thiran, J.-P. (2003). The BANCA database and evaluation protocol. In *4th International Conference on Audio- and Video-Based Biometric Person Authentication*, pages 625–638, Guilford, UK.

[7] Ballard, D. (1981). Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, **13**(2), 111–122.

[8] Belhumeur, P., Hespanha, J. P., and Kriegman, D. J. (1996). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. In *Fourth European Conference on Computer Vision*, pages 45–58, Cambridge, UK.

[9] Borgefors, G. (1988). Hierarchical chamfer matching: a parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence,*, **10**(6), 849–865.

[10] Butko, N. and Movellan, J. (2009). Optimal scanning for faster object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2751–2758.

[11] Chai, D. and Ngan, K. (1998). Locating facial region of a head-and-shoulders color image. In *Proceedings of the 3rd. International Conference on Face & Gesture Recognition*.

[12] Chetverikov, D. and Lerch, A. (1993). Multiresolution face detection. *Theoretical Foundations of Computer Vision*, **69**, 131–140.

[13] Christian, W., Gyuri, D., André, S., and Bernt, S. (2008). Sliding-windows for rapid object class localization: A parallel technique. In *Proceedings of the 30th DAGM symposium on Pattern Recognition*, pages 71–81.

[14] Colmenarez, A. and Huang, T. (1997). Face detection with information-based maximum discrimination. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[15] Crow, F. C. (1984). Summed-area tables for texture mapping. In *11th Annual Conference on Computer Graphics and Interactive Techniques*, pages 207–212.

[16] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893.

[17] David, S. and Richard, F. (1996). Toward robust skin identification in video images. In *Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition*.

[18] Dollár, P., Belongie, S., and Perona, P. (2010). The fastest pedestrian detector in the west. In *British Machine Vision Conference*.

[19] Extremal, M. S., Matas, J., Chum, O., Urban, M., and Pajdla, T. (2002). Robust wide baseline stereo from. In *British Machine Vision Conference*, pages 384–393.

[20] Fanelli, G., Gall, J., and Gool, L. J. V. (2011). Real time head pose estimation with random regression forests. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 617–624.

[21] Fen, Z. and Marcel, S. (2011). Evaluation of face detection and recognition. Idiap-Com Idiap-Internal-Com-01-2011, Idiap.

[22] Ferrari, V., Fevrier, L., Jurie, F., , and Schmid, C. (2008). Groups of adjacent contour segments for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **30**(1), 36–51.

[23] Fischler, M. and Elschlager, R. (1973). The representation and matching of pictorial structures. *IEEE Transactions on Computer*, **22**(1), 67–92.

[24] Fleuret, F. (2004). Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, **5**, 1531–1555.

[25] Fröba, B. and Ernst, A. (2004). Face detection with the modified census transform. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 91–96.

[26] Gall, J. and Lempitsky, V. (2009). Class-specific hough forests for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[27] Garcia, C. and Delakis, M. (2004). Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**(11), 1408–1423.

[28] Gavrila, D. M. and Munder, S. (2007). Multi-cue pedestrian detection and tracking from a moving vehicle. *International Journal of Computer Vision*, **73**, 41–59.

[29] Gross, R., Matthews, I., Cohn, J., Kanade, T., and Baker, S. (2008). Multi-pie. In *IEEE International Conference on Automatic Face Gesture Recognition*.

[30] Han, C.-C., Hong-Yuan Mark Liaoa, G.-J. Y., and Chenc, L.-H. (2000). Fast face detection via morphology-based pre-processing. *Pattern Recognition*, **33**(10), 1701–1712.

[31] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*.

[32] Hjelmas, E. and Low, B. K. (2001). Face detection: A survey. *Computer Vision and Image Understanding*, **83**, 236274.

[33] Hoogenboom, R. and Lew, M. (1996). Face detection using local maxima. *IEEE International Conference on Automatic Face and Gesture Recognition*.

[34] Hsu, R.-L., Abdel-mottaleb, M., and Jain, A. K. (2002). Face detection in color images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**, 696–706.

[35] hsuan Yang, M. and Ahuja, N. (1999). Gaussian mixture model for human skin color and its applications in image and video databases. In *Application in Image and Video Databases. Proceedings of SPIE 99*, pages 458–466.

[36] Huang, C., Ai, H., Wu, B., and Lao, S. (2004). Boosting nested cascade detector for multi-view face detection. In *International Conference on Pattern Recognition*, volume 2, pages 415–418.

[37] Huang, C., Ai, H., Li, Y., and Lao, S. (2007a). High-performance rotation invariant multiview face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**(4), 671–686.

[38] Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller, E. (2007b). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst.

[39] Jain, V. and Learned-Miller, E. (2010). Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst.

[40] Jamie, S., Andrew, B., and Roberto, C. (2008). Multiscale categorical object recognition using contour fragments. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, **30**, 1270–1281.

[41] Jesorsky, O., Kirchberg, K., and Frischholz, R. (2001). Robust face detection using the Hausdorff distance. In *International Conference on Audio- and Video-Based Biometric Person Authentication*, pages 90–95.

[42] Jones, M. and Viola, P. (2003). Fast multi-view face detection. Technical report, Merl TR2003-96.

[43] Jones, M. J. and Rehg, J. M. (1999). Statistical color models with application to skin detection. In *International Journal of Computer Vision*, pages 274–280.

[44] k Levi and Weiss, Y. (2004). Learning object detection from a small number of examples: the importance of good features. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 53–60.

[45] Ke, Y. and Sukthankar, R. (2004). Pca-sift: A more distinctive representation for local image descriptors. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 506–513.

[46] Kirby, M. and Sirovich, L. (1990). Application of the Karhunen-Loéve procedure for the characterisation of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**, 103–108.

[47] Lampert, C. H., Blaschko, M. B., and Hofmann, T. (2008). Beyond sliding windows: Object localization by efficient subwindow search. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

[48] Lehmann, A., Leibe, B., , and Gool, L. V. (2009). Feature-centric efficient subwindow search. In *IEEE Conference on Computer Vision*.

[49] Leibe, B., Leonardis, A., and Schiele, B. (2004). Combined object categorization and segmentation with an implicit shape model. In *Proceedings of the Workshop on Statistical Learning in Computer Vision*, pages 17–32, Prague, Czech Republic.

[50] Leo, B., H., F. J., A., O. R., and J, . S. C. (1984). *Classification and regression trees*. CHAPMAN and HALL/CRC.

[51] Leung, T., Burl, M., and Perona, P. (1995). Finding faces in cluttered scenes using random labeled graph matching. In *International Conference on Computer Vision*, pages 637–644.

[52] Li, J., Wang, T., and Zhang, Y. (2011). Face detection using surf cascade. In *ICCV workshop on Benchmarking Facial Image Analysis Technologies*.

[53] Li, S. Z. and Zhang, Z. (2004). Floatboost learning and statistical face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**(9), 1112–1123.

[54] Li, S. Z., Zhu, L., Zhang, Z., Blake, A., Zhang, H., and Shum, H. (2002). Statistical learning of multi-view face detection. In *Proceedings of the 7th European Conference on Computer Vision*, pages 67–81.

[55] Lienhart, R., Kuranov, A., and Pisarevsky, V. (2003). Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *25th Pattern Recognition Symposium*, pages 297–304, Madgeburg, Germany.

[56] Lifeng, H., Yuyan, C., Kenji, S., and Kesheng, W. (2009). Fast connected-component labeling. *Pattern Recognition*, **42**, 1977–1987.

[57] Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning.*, **2**(4), 285–318.

[58] Lowe, D. (1999). Object recognition from local scale-invariant features. In *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157.

[59] Lubomir, B. and Jonathan, B. (2005). Robust object detection via soft cascade. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 236–243.

[60] Ma, T. and Latecki, L. (2011). From partial shape matching through local deformation to robust global shape similarity for object detection. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1441–1448.

[61] Martinez, A. and Benavente, R. (1998). The AR Face Database. CVC Technical Report 24 Idiap-RR-49-2005.

[62] Messer, K., Matas, J., Kittler, J., Luettin, J., and Maitre, G. (1999). XM2VTSDB: The extended M2VTS database. In *Audio and Video-based Biometric Person Authentication*, pages 72–77.

[63] Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, **27**(10), 1615–1630.

[64] Mikolajczyk, K., Schmid, C., and Zisserman, A. (2004). Human detection based on a probabilistic assembly of robust part detectors. In *European Conference on Computer Vision*, pages 69–82.

[65] Moosmann, F., Triggs, B., and Jurie, F. (2007). Fast discriminative visual codebooks using randomized clustering forests. In *Advances in Neural Information Processing Systems*.

[66] Ojala, T., Pietikainen, M., and Harwood, D. (1996). A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, **29**(1), 51–59.

[67] Osuna, E., Freund, R., and Girosi, F. (1997). Training support vector machines: an application to face detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, page 130, Washington, DC, USA.

[68] Özuysal, M., Fua, P., and Lepetit, V. (2007). Fast keypoint recognition in ten lines of code. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, USA.

[69] Özuysal, M., Lepetit, V., and Fua, P. (2009). Pose estimation for category specific multiview object localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 778–785, Miami, USA.

[70] Pedersoli, M., Jordi, G., D, B. A., and J, V. J. (2010). Recursive coarse-to-fine localization for fast object detection. In *In European Conference on Computer Vision*, volume 6316, pages 280–293. Springer.

[71] Pedersoli, M., Vedaldi, A., and Gonzalez, J. (2011). A coarse-to-fine approach for fast deformable object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[72] Pedro, F. F., Ross, B. G., and David, A. M. (2010). Cascade object detection with deformable part models. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 2241–2248.

[73] Phillips, P., Moon, H., Rizvi, S., and Rauss., P. (2000). The feret evaluation methodology for face recognition algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(10), 1090 – 1104.

[74] Popoviciand, V., Thiran, J., Bailly-Bailliere, E., Bengio, S., Bimbot, F., Kittler, J., Matas, J., Ruiz, B., and Poiree, F. (2003). The banca database and evaluation protocol. In *4th International Conference on Audio- and Video-Based Biometric Person Authentication*, volume 2688, pages 625–638.

[75] Prasad, M., Zissermanand, A., Fitzgibbon, A. W., Kumar, M. P., and Torr, P. H. S. (2006). Learning class-specific edges for object detection and segmentation. In *Indian Conference on Computer Vision, Graphics and Image Processing*.

[76] Prisacariu, V. A. and Reid, I. (2009). Fasthog- a real-time gpu implementation of hog. Technical Report 2310/09.

[77] R., Z. and J., W. (1994). Non-parametric local transforms for computing visual correspondence. In *In European Conference on Computer Vision*, pages 151–158.

[78] Raphaël, F., Olivier, B., and Daniel, C. (1997). A constrained generative model applied to face detection. *Neural Processing Letters*, **5**(2), 11–19.

[79] Rodriguez, Y. (2006). *Face detection and verification using local binary patterns*. Ph.D. thesis, Ecole Polytechnique Fdrale de Lausanne.

[80] Rosten, E., Porter, R., and Drummond, T. (2010). Faster and better: A machine learning approach to corner detection. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, **32**, 105–119.

[81] Roth, D., Yang, M., and Ahuja, N. (2000). A SNoW-based face detector. *Advances in Neural Information Processing Systems 12*, pages 855–861.

[82] Rowley, H., Baluja, S., and Kanade, T. (1998a). Rotation invariant neural network-based face detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 38–44.

[83] Rowley, H. A., Baluja, S., and Kanade, T. (1995). Human face detection in visual scenes. Technical report.

[84] Rowley, H. A., Baluja, S., and Kanade, T. (1998b). Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(1), 23–38.

[85] Sakai, T., Nagao, M., and Kanade, T. (1972). Computer analysis and classification of photographs of human faces. In *First USA–Japan Computer Conference*, pages 55–62.

[86] Sauquet, T., Rodriguez, Y., and Marcel, S. (2005). Multiview Face Detection. Idiap-RR Idiap-RR-49-2005, IDIAP.

[87] Schneiderman, H. (2004). Feature-centric evaluation for efficient cascaded object detection. In *IEEE International Conference on Computer Vision and Pattern Recognition*.

[88] Schneiderman, H. and Kanade, T. (1998). Probabilistic modeling of local appearance and spatial relationships for object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[89] Schneiderman, H. and Kanade, T. (2000a). A statistical method for 3d object detection applied to faces and cars. In *IEEE International Conference on Computer Vision and Pattern Recognition*.

[90] Schneiderman, H. and Kanade, T. (2000b). A statistical model for 3d object detection applied to faces and cars. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 746–751.

[91] Shotton, J., Fitzgibbon, A. W., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1297–1304.

[92] Sirohey, S. (1993). Human face segmentation and identification. Technical Report CS-TR-3176, Univ. of Maryland.

[93] Subburaman, V. B. and Marcel, S. (2010a). An alternative scanning strategy to detect faces. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2122–2125.

[94] Subburaman, V. B. and Marcel, S. (2010b). Fast bounding box estimation based face detection. In *ECCV Workshop on Face Detection: Where we are and what next?*

[95] Sung, K. K. and Poggio, T. (1995). Learning a distribution-based face model for human face detection. In *Neural Networks for Signal Processing - Proceedings of the IEEE Workshop*, pages 398–406. IEEE, Piscataway, NJ, USA.

[96] Szarvas, M. Yoshizawa, A., Yamamoto, M., and Ogata, J. (2005). Pedestrian detection with convolutional neural networks. In *IEEE Intelligent Vehicles Symposium*, pages 224–229.

[97] Sznitman, R. and Jedynak, B. (2010). Active testing for face detection and localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32**, 1914–1920.

[98] Tae-Kyun, K. and Roberto, C. (2008). Mcboost: Multiple classifier boosting for perceptual co-clustering of images and visual features. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, pages 841–856. MIT Press.

[99] Taylor, S., Rosten, E., and Drummond, T. (2009). Robust feature matching in 2.3μs. In *IEEE CVPR Workshop on Feature Detectors and Descriptors: The State Of The Art and Beyond*.

[100] Terrillon, J. C., David, M., and Akamatsu, S. (1998). Automatic detection of human faces in natural scene images by use of a skin color model and of invariant moments. In *Proceedings of the 3rd. International Conference on Face & Gesture Recognition*.

[101] Tianyang, M., Xingwei, Y., and Jan, L. L. (2010). Boosting chamfer matching by learning chamfer distance normalization. In *Proceedings of the 11th European conference on Computer vision: Part V*, ECCV 2010, pages 450–463, Berlin, Heidelberg. Springer-Verlag.

[102] Torralba, A., Murphy, K. P., and Freeman, W. T. (2007). Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**(5), 854–869.

[103] Trefny, J. and Matas, J. (2010). Extended set of local binary patterns for rapid object detection. In *Computer Vision Winter Workshop 2010*.

[104] Turk, M. and Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, **3**(1), 71–86.

[105] Venu, G. (1996). Locating human faces in photographs. *International Journal of Computer Vision*, **19**, 129–146.

[106] Vincent, L. and Pascal, F. (2006). Keypoint recognition using randomized trees. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, **28**, 1465–1479.

[107] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518.

[108] Wu, B. and Nevatia, R. (2007). Cluster boosted tree classifier for multi-view, multi-pose object detection. In *IEEE 11th International Conference on Computer Vision*.

[109] Wu, B., Ai, H., Huang, C., and Lao, S. (2004). Fast rotation invariant multi-view face detection based on real adaboost. In *Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 79–84.

[110] Wu, J., Rehg, J., and Mullin, M. (2003). Learning a rare event detection cascade by direct feature selection. In *Advances in Neural Information Processing Systems*. MIT Press.

[111] Xiao, R., Zhu, H., Sun, H., and Tang, X. (2007). Dynamic cascades for face detection. In *IEEE International Conference on Computer Vision*.

[112] Yan, S., Shan, S., Chen, X., and Gao, W. (2008). Locally assembled binary (lab) feature with feature-centric cascade for fast and accurate face detection. In *IEEE International Conference on Computer Vision and Pattern Recognition*.

[113] Yang, G. and Huang, T. S. (1994). Human face detection in complex background. *Pattern Recognition*, **27**(1), 53–63.

[114] Yang, M.-H., Kriegman, D. J., and Ahuja, N. (2002). Detecting faces in images: A survey. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, **24**(1), 34–58.

[115] Yao, A., Gall, J., and Gool, L. J. V. (2010). A hough transform-based voting framework for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2061–2068.

[116] Yoav, F. and Robert, E. S. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37.

[117] Yow, K. and Cipolla, R. (1997). Feature-based human face detection. *Image and Vision Computing*, **15**(9), 713–735.

[118] Yow, K. C. and Cipolla, R. (1995). Towards an automatic human face localization system. In *Proceedings of 6th British Machine Vision Conference*, pages 701–710. Springer–Verlag.

[119] Zhang, C. and Zhang, Z. (2010a). A survey of recent advances in face detection. Technical report.

[120] Zhang, C. and Zhang, Z. (2010b). Winner-take-all multiple category boosting for multi-view face detection. In *ECCV Workshop on Face Detection: Where are we, and what next*.

[121] Zhang, H., Gao, W., Chen, X., and Zhao, S. S. D. (2006). Robust multi-view face detection using error correcting output codes. In *Proceedings of the European Conference on Computer Vision*, volume 4, pages 1–12.

[122] Zhang, L., Chu, R., Xiang, S., Liao, S., and Li, S. Z. (2007a). Face detection based on multi-block lbp representation. In *Lecture Notes in Computer Science, Advances in Biometrics*, volume 4642, pages 11–18, Berlin. Springer.

[123] Zhang, W., Zelinsky, G., and Samaras, D. (2007b). Real-time accurate object detection using multiple resolutions. In *IEEE 11th International Conference on Computer Vision*.

[124] Zhou, S. K. and Comaniciu, D. (2007). Shape regression machine. In *Information Processing in Medical Imaging (IPMI)*.

Curriculum Vitae

Name: Bala Subburaman Venkatesh

Nationality: Indian

Email: venkatesh.bs@gmail.com

General area of interest:

Computer vision, pattern recognition, deep learning, robotics

---

Education:

1. *September 2007 - February 2012*

   PhD in Electrical Engineering at Ecole Polytechnique Fédérale de Lausanne, Switzerland and the Idiap Research Institute, Martigny under the supervision of Prof. Hervé Bourlard, and Dr. Sébastien Marcel (defense scheduled: 14th February 2012).

2. *August 2000 - September 2003*

   Master of Science by Research in Computer Science and Engineering, at Indian Institute of Technology Madras (IITM), Chennai, India, under the supervision of Prof B. Yegnanarayana. (CGPA: 9.0/10).

3. *September 1995 - September 1999*

   Bachelor of Engineering in Electronics and Communication Engineering at M. S. Ramaiah Institute of Technology, Bangalore, India (First class).

Professional Experience:

1. *September 2007 - March 2012*

   Research Assistant at the Idiap Research Institute, Martigny, Switzerland.
2. *July 2011 - October 2011*

   Internship at MultiTel, Mons, Belgium with Cyril Carincotte
3. *April 2005 - August 2007*

   Engineer, General Motors, Technical Center India. (Onsite - International Technical Development Center, Adam Opel AG, Germany, Sept. 05 - Feb. 06, Sept. - Nov. 06)
4. *December 2004 - March 2005*

   Research Fellow, John F. Welch Technology Centre (JFWTC), General Electric, Bangalore, India.
5. *October 2004 - November 2004*

   Project Associate, Super Computer Education Research Center, Indian Institute of Science, Bangalore (IISc), India.
6. *October 2003-June 2004*

   Project Assistant, Department of Electrical Engineering, Indian Institute of Science, Bangalore, India.
7. *February 2000 - June 2000*

   Project Assistant, Center for Electronics Design and Technology, Indian Institute of Science, Bangalore, India.

List of Publications:

Conference papers (peer-reviewd):

1. B. S. Venkatesh, and Sebastien Marcel, "Fast Bounding Box Estimation based Face Detection," Workshop on Face Detection: Where we are, and what next? (In conjunction with ECCV 2010), Greece, 2010. (**Best paper award sponsored by Microsoft Research India**)
2. B. S. Venkatesh, and Sebastien Marcel, "An Alternative Scanning Strategy to Detect Faces," IEEE International Conference on on Acoustics, Speech, and Signal Processing (ICASSP), Dallas, Texas, 2010.

3. A. Jaya Kumar, B. S. Venkatesh, and Y. V. Venkatesh, “Stereo disparity estimation using supervised neural network,” IEEE International Workshop on Machine Learning and Signal Processing, Brazil, 2004.
4. S. Palanivel, B. S. Venkatesh, and B. Yegnanarayana, “Real time face authentication system using autoassociative neural network models,” IEEE International Conference on Multimedia and Expo, Baltimore, USA, July 2003, pp. 257-260.
5. S. Palanivel, B. S. Venkatesh, and B. Yegnanarayana, “Real time face recognition system using autoassociative neural network models,” IEEE International Conference on Acoustics, Speech and Signal Processing, Hong Kong, April 2003, pp. 833-836.
6. B. S. Venkatesh, S. Palanivel, and B. Yegnanarayana, “Face detection and recognition in an image sequence using eigenedginess,” Indian Conference on Computer Vision, Graphics and Image Processing, Ahmadabad, India, December 16-18, 2002, pp. 97-101.

Technical Reports different from above:

1. Bala Subburaman Venkatesh and Sebastien Marcel, “Face Detection using Ferns,” https://publications.idiap.ch/index.php/publications/show/1575, Tech. Rep., Idiap-Internal-Com-06-2008.
2. Bala Subburaman Venkatesh and Sebastien Marcel, “A Fast Search Technique for Face Detection using Location Estimation”, (Journal submission), Tech. Rep., Idiap-Internal-RR-171-2010.