RHEINISCH-WESTFÄLISCHE TECHNISCHE
HOCHSCHULE AACHEN
KNOWLEDGE-BASED SYSTEMS GROUP
PROF. GERHARD LAKEMEYER, PH. D.

Detection and Recognition of Human Faces using Random Forests for a Mobile Robot

MASTER OF SCIENCE THESIS

VAISHAK BELLE
MATRICULATION NUMBER: 26 86 51

SUPERVISOR:	PROF. GERHARD LAKEMEYER, PH. D.
SECOND SUPERVISOR:	PROF. ENRICO BLANZIERI, PH. D.
ADVISERS:	STEFAN SCHIFFER, THOMAS DESELAERS

Declaration

I declare that this Master Thesis is my own unaided work, except for the official assistance from my supervisors. It has not been submitted in any form for any other degree or diploma at any university or other education institution. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is included at the end.

Aachen, March 27, 2008

Acknowledgements

I would first like to thank Prof. Gerhard Lakemeyer for an opportunity to work with the Knowledge-based Systems Group. I would then like to express my sincere gratitude to both Stefan Schiffer and Thomas Deselaers for an incalculable number of suggestions, critical talk and motivational sessions. I owe the entire direction of my thesis to the both of them. I would like to specially mention the utmost care S. Schiffer has taken on necessary corrections to my proposal and the final outcome and general advice he has offered over the last 8 months.

I would like to thank the AllemaniACs RoboCup team for a great work environment and putting up with me in general, especially at lunchtimes. I would also like to acknowledge Tim Niemueller for proof-reading the last drafts of the thesis and his subsequent suggestions. I thank my friends here in Aachen, who were almost a surrogate family and patiently lent their ears to my complaints.

Lastly and most importantly, I dedicate this thesis to my mom, who has been the source of constant encouragement. At no point do I except to be able to completely express the gratitude I feel towards her.

CONTENTS

1 Introduction

In this master thesis, we present a new framework for the automatic detection, recognition and learning of human faces with random forests. The principal motivation for our work is towards the design of a fast face processing system, especially applicable to mobile robots. Given a high amount of social interaction between a robot and human beings, the experience is rendered more *natural* if the robot is able to detect faces and recognize corresponding identities. If the identity of the person is not known then it is *learnt* by the robot.

1.1 Motivation

Robots that interact with humans, such as rescue robots and service robots, frequently encounter human faces. Rescue robots are principally designed to be of aid to rescue workers and humans in danger. Typically scenarios where rescue robots are deployed include disaster-struck areas, explosions, infernos and building collapses. The robots self navigate in hazardous environments to either probe for surviving humans or to neutralize any immediate danger and make the surrounding less perilous for human rescue workers.

Service robots, generally, are designed to offer assistance to humans and to people with disabilities, in particular. They could be used for day to day activities, such as retrieving tools and objects, for and by humans (civil usage). They could also be used to provide feedback on approaching entities to the visually impaired and in general, provide structural support for the physically or mentally impaired.

In both of the applications of robotics discussed above, there is a high amount of *social interaction* between the robots and the human beings. The robots respond dynamically to requests and communicate with human beings. An indispensable characteristic of such robots is the automatic detection and recognition of the faces of humans they interact with. To elucidate further, the interaction will be rendered more human-like if a robot is able to identify a human from a group of *known* individuals. Further, if the identification fails then the robot may introduce itself like any of us. It must then learn the new individual and assimilate the identity into its recognition system. We propose a face processing system that targets the aforementioned requirements.

1.2 Approach and Contributions

Central to our approach are random forests. Random forests are defined as "a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all the trees in the forest" [1]. In our thesis, we present an integrated framework to the detection, recognition and learning of human faces using random forests. Such a system will find immediate applications in rescue and service robotics as outlined above.

As emphasized, the focal point of our work is towards an improved human-robot interaction. An important criterion for a prospective face recognition system is *response time*. A robot that reciprocates with a massive delay, at least in relative comparison to human beings, will deliver a frustrating experience. A robot that requires an extensive period of time to learn and recognize people is an antithesis to our expectations. It is thus clear that service robots that accompany the elderly and disabled are to be of assistance to their owners just as human companions would. It is of essence to have a fast face processing system in place.

In line with our contribution to rescue and service robotics, we aim to be applicable to *RoboCup@Home*. RoboCup@Home is "the largest home robotics competition" and is held annually [2]. This competition is a division of the RoboCup competitions that "attempt to foster AI and intelligent robotics" [3]. RoboCup momentarily encompasses an eclectic collection of games, many of which encourage cooperation between a machine and its human counterpart [4]. The robots are expected to work autonomously. Zant and Wisspeintner [3] discuss tests that outline a few application scenarios. In some of the scenarios, owner identification is predominant. Learning of new identities present an added improvement in the behavioral component of the robot. Our proposal, which presents an integrated framework to detection, recognition and learning of new faces will bring the RoboCup robots closer to the point of succeeding in the above evaluations.

Contributions to robotics aside, the framework presents an addition to the recent applications of random forests as a statistical learning procedure. Random forests have been applied, with promising results, in image classification [5, 6, 7, 8, 9, 10], image matching [11], positioning discriminatory and generative image regions [12], segmentation [13] and gesture recognition [14]. Extending the literature above to develop random forests for face detection and recognition presents an interesting research goal. Random forests are proven to be fast [12, 5, 6, 7, 8, 9, 10, 15] which is pertinent with our motivation. Further, a random forest is a powerful statistical framework [1] with a very high generalization accuracy. A good generalization accuracy is critical for many learning algorithms, in general, and image classification in particular as it remarks on the performance of the algorithm on novel data. Image classification tasks encounter substantial variations in the image data such as occlusions, illumination variations

and object pose changes. Our framework effectively casts face detection and recognition as a classification task and hence the aforementioned theoretical benefit of a high generalization accuracy is significant.

In general, random forests present the following advantages:

1. The approach is generic. It is easily extendable to the detection and recognition of any object other than faces.
2. Image classification is reported to be successful in spite of partial occlusions [16].
3. Random forests are a parallel learning algorithm - critical components can be executed completely in parallel.
4. The approach has a quick training phase.
5. Random forests can achieve an extremely high generalization accuracy.
6. The nature of random forests presents a uniform strategy for accomplishing many image classification tasks.

In conclusion, our goal is to build a fast and unified framework for face detection, face recognition and face learning using random forests for mobile robots.

1.3 Outline

The outline of the thesis is as follows: Chapter 2 discusses literature on the applications of random forests for image classification, object detection, segmentation and gesture recognition. Chapter 3 reviews the general framework for any face recognition system and enumerates approaches to face detection. Chapter 4 presents feature extraction, face recognition and publicly available training data for face detection and recognition[1]. Chapter 5 discusses our framework: face detection, recognition and learning using random forests. Chapter 6 presents comparative evaluations on our approach. We draw our conclusions and outline future work in Chapter 7.

[1] Although our organization of related work in Chapter 2 is limited to applications of random forests, it should be noted that the face detection and face recognition approaches reviewed are also relevant as related work. Chapter 2 only presents frameworks analogous to our own.

2 Related Work

In this chapter, we review literature that use random forests for a number of related image classification tasks. The approaches enumerated below, in addition to our work, grow decision trees by randomizing the decision criteria. In Section 2.1, we introduce a few basic concepts of a learning framework followed by an overview of classical decision trees and random forests.

The literature review, in Section 2.2-2.5, covers applications of random forests to image classification, object detection, segmentation and gesture recognition. The applications additionally reveal some of the advantages of random forests. We conclude this review in Section 2.6.

2.1 Basic Concepts

A supervised learning system is characterized by a learning algorithm and labeled training data [17]. The algorithm dictates a process of learning from information extracted, usually as *feature vectors*, from the training data.

Training data is typically of the form:

$$D = \left\{ \left(\Gamma_n, c_p \right) : n = 1, \ldots, N; p = 1, \ldots, P \right\} \tag{2.1}$$

where Γ_1^N are the data and c_1^P are the corresponding class labels. If the training data are images then the data is considered to be M dimensional, i.e. the images are defined by $K \times L$ pixels (where $K * L = M$). As we work only with images in this work, we limit our discussions to the same.

Any rectangular region of an image is a local characteristic of the image and is referred to as a local visual descriptor [12]. As images, in general, are high dimensional the descriptors are quantized as feature vectors using filters. **Feature vectors** are a more efficient representation for purposes of comparisons and computations by the learning framework.

A general description of a **filter** is given by,

$$f(\Gamma) : \Gamma \mapsto \Re^{M'} \tag{2.2}$$

where $M' < M$ and $\Re$ is the space of real numbers. Essentially, a filter maps an image or a region to either a scalar ($M' = 1$) or a vector (characterized by dimensions lower than M). The resulting entity is termed as a feature or a feature vector respectively.

A learning framework uses the feature vectors extracted from the labeled training data to build a *classifier* for classification. The task of classification is to assign a label on novel data given labeled training data.

A general definition of a **classifier** is given by,

$$h(\Gamma) : \Gamma \mapsto p \tag{2.3}$$

where $p = \{1, \ldots, P\}$ is the class label awarded to the novel image Γ. To investigate the performance of a learning framework, we use a labeled *test data* collection that measures the accuracy of the labels awarded. Formally, classifier accuracy is probed with a **correctness indicator function** $I_h(\Gamma)$ defined as,

$$I_h(\Gamma) = \begin{cases} 1 & \text{if } h(\Gamma) \text{ is the true label of the test image } \Gamma \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

2.1.1 Classical Decision Trees and Random Forests

A random tree is structurally homogenous to a classical decision tree. Classical decision trees and random trees are grown recursively top down. Classical decision trees find a *decision criterion* that best divides the multi-class training data into two groups at each node [18]. A decision criterion consists of an attribute and a related threshold.

Random trees, on the other hand, randomize the decision criteria. A random decision criterion is defined by a random attribute and a random threshold. Random trees, thus, divide the training data on completely random attributes. This randomization mainly addresses high dimensional data and generalization [19, 20, 1]. To elucidate, classical decision trees perform poorly when trained on high dimensional data, such as images, as an exhaustive search to find the best split criterion is computationally expensive. As an example, a 50×50 image is 2500 dimensional. Classical decision trees also suffer from overfitting or low generalization with high dimensional data [21, 22, 18]. Overfitting is the condition of a grown decision tree that is meticulously consistent with the the training data and yet not learnt "useful" information. The tree has, in this case, maximized information gain on non-critical attributes due to the high dimensionality of the training data and the attributes are too general to be of any use. The performance, with overfitting, is low on *unseen* or *novel* data. Image data has substantial variations due occlusions, illumination and pose changes and hence, a high generalization accuracy is important. Random trees thus, address two of the shortcomings of classical decision trees that are especially applicable to image training data.

To reiterate our description, a random tree selects a random *feature* and a random threshold and divides the training images at a node. Recursively, a complete tree and subsequently a collection of trees are built that constitute a random forest. The classification results from each tree are collected for an input image and typically, a simple majority voting scheme awards the resulting class label. In the following sections, we present the use of random forests on a number of related image classification tasks.

2.2 Image Classification

Random forests for image classification have been extensively covered by Marée et al [5, 6, 9, 10]. A general description of the image classification problem is to *categorize* a previously unseen input image and subsequently assign a label using information drawn from labeled training images [10]. Image classification is used with images from geology, biology, building images, cars, motorbikes etc. for various labeling tasks.

The central idea of the approach taken by Marée et al. [10] is to sample random subwindows, scale them and grow a random forest from features extracted from the subwindows. The sampled subwindows are random both in their location and size. The size is allowed to range between 1×1 pixels to the maximum square window possible. Once sampled, the subwindows are rescaled to 16×16 pixels.

An illustration ensues in Figure 2.1. Here, numerous subwindows, of different sizes, are sampled and account for decisions at different levels of a tree. The textures in the figure represent local patterns. Three random trees are shown, each of which is built independently. The trees necessarily need to be built independently as the generalization error is inversely proportional to the independence of the trees [1].

Classification on an input image is carried out by randomly extracting subwindows in the image and propagating scaled subwindows down each tree in the forest. Votes from each of the random trees for all sampled subwindows are aggregated. A simple voting scheme with majority ruling awards the corresponding class label. Experimental evaluations on a variety of image classification databases, namely (i) COIL-100, a dataset of 100 different 3D objects, and (ii) ZuBuD, a dataset of 201 buildings in Zürich were presented. The results are very competitive.

Marée et al. [9] discuss that one of the benefits of the random forest approach is that the classification is robust to partial occlusions. This is attributed to the fact that not all the sampled subwindows of the input image need to be correctly labeled. The voting scheme allows the classifier sufficient flexibility to falsely categorize a few subwindows.

Marée et al. [5, 10] also claim that the rescaling of all sampled subwindows to a fixed dimension for training renders the classifier scale invariant. Further, the pixel values are moved to the Hue-Saturation-Value (HSV) space. The HSV value considers the

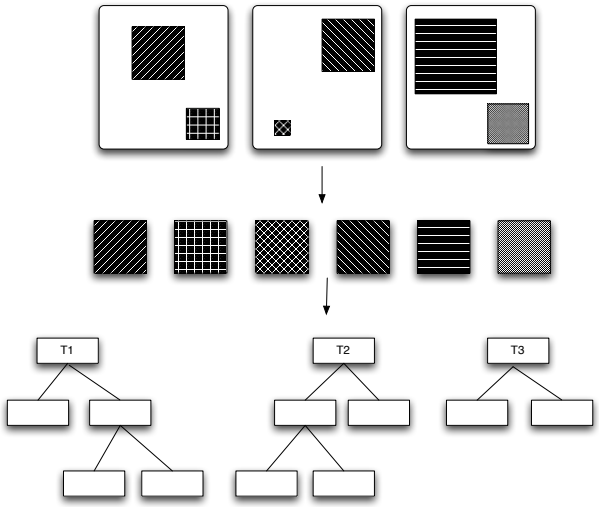

Figure 2.1: Random forest from random subwindow sampling and scaling. The subwindows are scaled to the 16×16 (see text). Trees $T1$, $T2$ and $T3$ are grown independently.

image in terms of its color (hue), vibrance (saturation) and brightness (value). The significance of images in the HSV space is that the illumination is shifted to only one band (brightness) when compared with three separate bands in the RGB color space. The input from the brightness band is limited for all training and input images which renders the classifier illumination invariant.

An increase in the classification accuracy is noted on using more than one tree. Marée et al. [10] report that, with extra trees, the error rate is reduced by more than 6% on the COIL-100 protocol.

The above classifier used simple voting to aggregate results from all the trees in the forest. A more complex aggregation mechanism is the use of *visual codebooks*. Moosman et al. [12] sample the training images, quantize the descriptor vectors and grow a random forest as above. Each leaf node is awarded a distinct *visual word* from the features present in the node. An input image is sampled and propagated down each random tree to the leaf nodes. The visual words of the corresponding leaf nodes are histogram plotted. A Support Vector Machine (SVM)[1] on the histograms identifies the class label for the image. Thus, the trees function on a spatial partitioning mechanism [12]. The approach shares similarity to the *bag-of-words* model [23] from text classification where a collection of cues, not necessarily ordered, are used for classification.

[1] We review SVM in Section 4.2 and Figure 4.3.

2.3 Object Detection

Image matching extends image classification and can be used to detect "objects" in a background. The problem can be defined as follows: given knowledge of the geometric appearance of an object, followed by a few scenes of the object in a background, object detection identifies the extent and the position of objects, if any, in an input image [16]. The presence of objects in the input image is as determined by a human.

Figure 2.2: Selection of a few hundred keypoints in an object [11].

Lepetit et al. [11] present a selection of 200 keypoints based on a probability measure. Figure 2.2 shows the selection of such keypoints in a stuffed toy. A random tree is grown by creating tests at nodes of the kind: is the intensity of the pixel at (x, y) higher than the intensity of the pixel at $(x + 1, y)$? Forests are grown by using random subsets of the training images for each tree to ensure increased independence between the trees. To detect the object, patches of the input image are propagated down the tree and keypoints are matched.

2.4 Segmentation

Yin et. al [13] present a random forest based monocular segmentation approach. The principal application targeted is video chat with background substitution. Segmentation is defined as the separation of an object of interest from its background. While the most competitive results to segmentation have been achieved with stereo vision [24], comparable results are presented using monocular vision and random forests.

Spatio-temporal derivates are used as "motion" filters. The filters are clustered and titled as *motons*. Each moton represents a unique response of input images (images from the video camera) to motion analysis. For instance, image regions that "move" in the background are identified with clustered spatio-temporal derivates as the "moving edge" moton. Regions of an image with weak textures are identified with a "weak

texture" moton. Regions of the image that do not move are identified with a "stationary-edge" moton.

Rectangular regions sampled from captured still images are paired with motons as *shape filters*. The shape filters are used in training random forests. A subset of shape filters are used to train each tree. Training with subsets of shape filters guarantees independence between the random trees. An approach previously discussed in Section 2.3 also train random trees with subsets to ensure independence.

When the video chat is initiated, a captured image is propagated down each tree. The shape filters segment the foreground from the background. The latter is then substituted with any other image.

2.5 Gesture Recognition

Gesture recognition capture and identify human gestures. Hand gestures are used to accomplish tasks such as copy and paste, erase selected text and move windows just as one would with a real object. Deselaers et al. [14] use stereo vision to capture depth and present a unified framework for recognition of object classes (hands holding objects), gesture recognition and touch events, using random forests.

Manually segmented hand gestures (positive samples) and background images such as tables and desks (negative samples) serve as the training data for the random forests. The decision criteria is characterized by the comparison of pixel intensities at random locations in the training data with randomly chosen thresholds. The initial task of the framework is the segmentation of the hand region from the background in a captured still image. The captured image is propagated down a forest trained as above and the framework segments the hand from any background.

To recognize the gesture of the segmented hand, each tree is grown further with multi-class hand gesture training data. Here, each gesture is a class and the training data is composed of stereotypical images for this gesture. Propagating the same down the extended tree will now recognize the gesture. To recognize touch events, the trees are extended further with stereotypical touch event image as training data.

Figure 2.3, for instance, depicts a few hand positions. The images stereotype various object classes, gestures and touch events. The images are the training data for the three separate recognition tasks modeled in a single integrated framework.

Figure 2.3: A set of sample hand positions stereotypical of clutching objects (pen, paper, card) and actions (grab, select). [14].

2.6 Summary

As discussed previously, the approaches are, in principle, analogous image classification tasks. The exact nature of the training data used is dependent on the specific goal of the learning framework. The decision criterion in every approach enumerated in this review is characterized by a random component. Marée et al. [9, 5, 6, 10] and Moosmann [12] sample rectangles of random dimension at random locations. Lepetit et al. [11] sample pixels at random locations. Yin et al. [13] randomly sample shape filters, defined as clustered spatio-temporal operators paired with rectangular regions. Deselaers et al. [14] sample pixels at random locations. Subsequent to the growth of the forest, the classification tasks sample the input images and propagate the same on the trees to return class labels. In our own framework, the random component is the sampling of randomly sized rectangles at random locations from the training data. The framework is presented ensuing a review of the current approaches to face detection and face recognition.

3 Face Detection

As Yang et al. [25] discuss, research in face recognition proceeds with the assumption that the face is already segmented from the background image. However, a complete face processing system must include an additional face detection system that is able to segment or extract faces from a complex background and supply the faces to a recognition module.

An intermediary stage is that of extracting features. As many face detection approaches train on features, feature extraction may be an implicit component in the system. Figure 3.1 shows the three stages of face recognition. In this chapter, we discuss approaches to face detection.

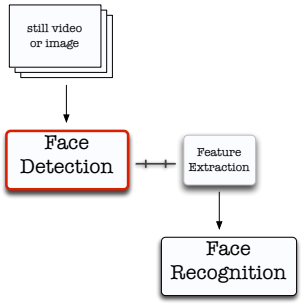

Figure 3.1: The three stages of a face processing system.

3.1 Introduction

Face recognition systems are currently one of the most important and widely used applications of image understanding and pattern recognition [26]. Some common applications include,

- **Service and Rescue Robots:** offer assistance, neutralize perilous environments, structural support to the disabled.
- **Biometrics:** driver's license, passports and voter registrations.
- **Information Security:** file encryption and desktop logon.
- **Law Enforcement and Surveillance:** CCTV cameras and suspect tracking.

- **Access Control:** facility access (access to buildings and houses).

In the above, face images serve as the primary input to accomplish tasks of encryption and identification. We note that applications such as suspect tracking might have to deal with different head gaits, partial occlusion of the faces due to foreign objects among others. Even a desktop logon system can expect the presence of a foreign body in the background as can a facility access provider. None of these applications can thus straightforwardly carry out face recognition. The crucial step is to precisely locate the faces in the image prior to further processing. This step is referred to as face detection. Face detection is thus defined as, "Given an arbitrary image, the goal of face detection is to determine whether or not there are any faces in the image and, if present, return the image location and extent of each face" [25].

A face detection system, operating in real life scenarios, can be expected to encounter any of the challenges listed below. For instance, a service robot and its supporting camera usually captures furniture, different light sources, other objects along with humans.

- **Pose variations.** Excluding the sole case of a human facing the camera either out of participation or accident, most applications can expect different side profiles. Some applications, such as a smaller robot interacting with a human, capture still images from a lower angle. CCTV Cameras usually capture human faces from an upper angle. In rare cases, upside down and rotated faces are also captured.
- **Occlusions.** Foreign objects obstructing the view of the camera make the task of detection (and recognition) difficult. Depending on the percentage of the face that is occluded, face detection may even be impossible.
- **Illumination.** Another challenge is related to the illumination conditions under which the image was obtained. For instance, if the illumination was very low, the face contour might not be discernible from the background. Only a part of the face may be illuminated. Further, the source and direction of the illumination is also important.
- **Facial components and structures.** The presence or absence of beards and mustaches, changed hair styles are also challenges.

In most real life scenarios, a captured image includes multiple faces. A simplified problem is, thus, to assume the presence of only one human face in the input image. This is referred to as the *face localization* problem. An alternative is the task of locating features, such as eyes, nose, ears and mouth, and is referred to as *facial feature detection*.

A measure of the effectiveness of a face detection system is the *detection rate*. The detection rate is calculated as,

$$\text{Detection rate} = \frac{\text{Number of faces correctly detected}}{\text{Total number of faces in the image as determined by a human}}$$

where the rate is derived from the generic correctness indicator function previously defined in Equation 2.4.

A detection stage can commit two kinds of errors: *false positives* and *false negatives* [25]. False positives are referred to as those errors when a face was reportedly detected by the system but there is no face in the image. A false negative is recorded in case a face is not detected, i.e. a region of the input image is falsely assumed to not contain a face. Further, awarding the "bounding box" returned to be a correctly detected face is determined by constrains that declare the percentage of overlap between the returned bounding box and the actual face as a threshold. Additionally, the extent of the face (area of the bounding box) independently is measured to the extent as determined by a human being. If the bounding box is too large or too small, then the result is deemed as invalid.

The granularity of false positives and negatives can be reviewed further with the *alignment error*. The alignment error is the case of a subwindow nominated as a face, but the position and extent of the detection only partly covers the face. Depending on the precision expected of the detections, the alignment errors border false positives or false negatives. We note that procuring very high detection rates with an increasing number of false positives is not a fair evaluation, i.e. the false positives count is almost as important as the detection rate. Algorithms search for a tradeoff between the detection rate and the number of false positives. A graphical tool that monitors this tradeoff is the Receiver Operating Characteristics (ROC) curve [27, 28].

It can be concluded from the above that face detection, as a machine learning approach, is a two class problem: the task is to identify if an image region is a “face” or a “non-face”. In accordance, the training data for face detection is organized into two collections: faces and background images. In the following section, we review a few approaches to face detection.

3.2 Approaches To Face Detection

Traditionally, the approaches to face detection can be roughly divided into four categories: knowledge-based, feature-based, template matching and appearance-based. Recent literature [29, 30], including our work, utilize a generic sampling-based approach. We review all five categories in this section.

3.2.1 Knowledge-based methods

Knowledge-based methods extrapolate the human understanding of the structural characteristics of a face. Rules are formalized from morphological facts. The presence of two eyes which are symmetrical, a triangular nose, the relative difference in color

between parts of the human face etc. are encoded as relationships. If the rules encoded are too specific, the detection accuracy drops as small variations in the requirements results in false negatives. If the rules are too general, many false positives result. To allow for variations in the structural descriptions, a few approaches employ fuzzy theory.

The approach typically builds on multi-resolution or *mosaic* images. Initially, image analysis is carried out at a very low resolution to probe for any high level morphological description. Subsequently, the resolution is increased and the descriptions are more specific.

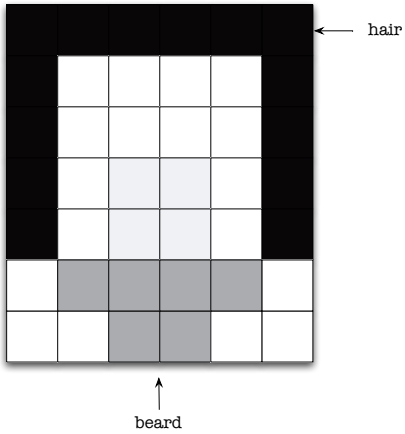

Figure 3.2: Structural characteristics are encoded as rules in different cells. The diagram depicts a Π shaped hair region, a beard region and central homogenous nose and mouth region [31].

Yang and Huang [32] divide a stereotypical face into a number of units or *cells* that must conform to specified descriptions as shown in Figure 3.2. In the figure the central region of the face, for instance, occupy 4×4 cells. This is referred to as a quartet. The bottom portion of this quartet occupies 4 cells (light gray colored) and typically, this region of a face is homogeneously colored. Thus, a rule that applies here is "this part of the face has four cells with a basically uniform intensity". Similarly, a Π-shaped region composed of dark pixels and of significant length denotes a stereotypical hair area. A stereotypical beard description is depicted in the figure as well.

At a higher resolution, the quartet is subdivided into 2×2 regions or octets. With an analogous rule description, the eyes and nose are detected. The approach is thus a hierarchical knowledge-based recognition system [32, 31, 25]. Kotropoulos and Pitas [31] extend the work by reducing computational needs and a preprocessing step to estimate cell dimensions.

The importance of this technique is the modeling of human intuition although it suffers from being more of a localization technique as opposed to detection of multiple faces.

3.2.2 Feature-based methods

Feature-based methods are those that take a bottom-up approach by locating facial features initially and then collecting their respective enclosing entities such as edges, blobs, streaks and graphs as a detected face. Typically, edge detectors are used to identify particular shapes such as eyebrows, eyes, noses etc. and statistical models estimate distances between these shapes. A linking stage follows that collects them into groups and subsequently detects a face. Figure 3.3 illustrates this approach: edge filtering and grouping to detect faces.

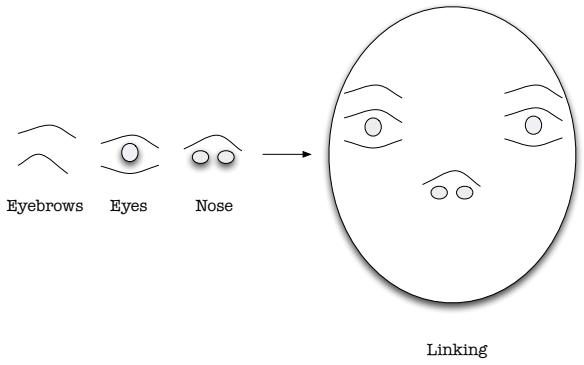

Figure 3.3: Edge detection followed by linking and grouping to detect faces.

Sirohey [33] reasons that since the shape of a face is roughly elliptical, facial features can be combined in an ellipse fitting probe. Essentially, he uses a Canny Edge Detector[1] [34] to identify all the edges. Heuristics are used to create an edge map, i.e. unconnected edges (the output of the edge detector) are linked to verify if ellipses formed thus conform to that of a face contour. The final result is a detected face.

While Sirohey uses edges of face features and face contours, any sort of morphological identity can be chosen. For instance, learning the texture of skin and hair is used to detect faces [25]. The use of skin color in color spaces, such as RGB and HSV, to accomplish detection is also reported [35].

Graf et al. [36] use a combination of features to detect faces from videos. Their approach uses shape modeling, color response and motion. The first stage uses spatial frequency filters to identify the best set of facial features (eyes, nose etc.) that can be used to

[1] A Canny Edge Detector is an multi-stage detection technique. A complete description of the technique is beyond the scope of this thesis.

detect faces. The second stage trains on the best thresholds for responses to color filters that can segment faces from the background. The last stage calculates the absolute difference in regions to identify "moving" pixels to segment the human face. This is similar to a spatio-temporal filter. The three different stages supply a list, as cues, and a *n-gram*[2] model decides on the location and extent of a face. Feature-based approaches, in general, perform poorly with blurred images [25].

3.2.3 Template Matching

Template matching is based on correlation values. Features such as contours and edges are extracted and their relative locations are matched against predetermined estimates. Conformance to a certain 'skeleton' is considered to be a detection.

Scassellati [37] uses *ratio templates* for quick detection of faces. He considers ratio templates to be "biologically plausible" as the templates can be hand coded or learnt from training data, much like the learning models in humans. The approach virtually divides the human face into 16 regions of interest using a 14×16 pixels grayscale window. Each region is averaged using a grayscale window. Modeling the typical relative brightness between the regions, the ratio template outlines constrains for a region to be considered to contain a face. The regions also intuitively denote different portions of a face, such as the temple (left or right) and forehead (left or right). Typically, irrespective of the illumination conditions, the eyes are darker, the nose is homogeneously brighter etc. thus making ratio templates illumination invariant. Figure 3.4 illustrates the divisions and relationships in the model. As faces in an input image can occur at many scales, the grayscale template is probed for all possible regions of the input image at different scales to detect faces.

An improvement to the "fixed" templates, such as the approach above, are *deformable templates*. With deformable templates, sufficient flexibility in the template structure is allowed so as to account for variations and thus increase the detection rate. They address some of the shortcomings of fixed templates where the latter suffers from low detection rates with shape, scale and pose variance.

Yuille et al. [38] propose a generic deformable template based on an energy minimization function. The energy function is defined by peaks, edges and valleys corresponding to a feature. For instance, they define an eye template characterized by four properties, the first of which is a circle of an specific radius that corresponds to the enclosing circle of the iris. The second describes a contour for the eye. The third and fourth model the whites of the eye. The template definitions are built as a function and the algorithm searches for a conformance to the definitions, i.e. lowest value that

[2] An n-gram is a probability measure for subsequences. Here, sequences of features from the three stages that lead to a detected face in the training data are learnt.

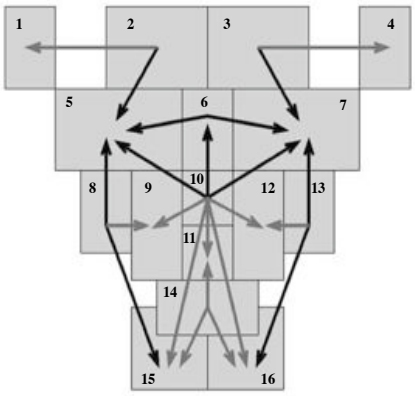

Figure 3.4: A division of a stereotypical face into 16 regions of interest. The face is assumed to be 14×16 pixels in dimension. Relationships are encoded to detect faces [37, 25].

the function can achieve, in the input image. Such templates can be extended to other features and the face itself.

3.2.4 Appearance-based models

Appearance-based models attempt to reduce the dimension of the training data before extracting statistical properties. Typically, these models calculate the class-conditional probability for a feature vector extracted from the input image. If the feature vector is x, then we calculate the Bayesian likelihoods $p(x|face)$ and $p(x|non\text{-}face)$ from the training data [25]. Since a direct computation of likelihoods is computationally expensive techniques such as Principal Component Analysis (PCA) are used for dimensionality reduction.

Eigenfaces [39, 40] is a pioneering appearance-based approach [41]. Let us consider M dimensional training data denoted as Γ_1^N. PCA attempts to find an effective and efficient representation of the training data. From the training data $\Gamma_1, \ldots, \Gamma_N$, the average image and the covariance matrix is calculated as,

$$\psi = \frac{1}{N} \sum_{n=1}^{N} \Gamma_n$$

$$C = \frac{1}{N} \sum_{n=1}^{N} (\Gamma_n - \psi)(\Gamma_n - \psi)^T \tag{3.1}$$

where ψ is the average and C is the covariance.

The eigensolution for the covariance matrix is calculated and the eigenvectors are ordered in decreasing eigenvalues. The eigenvectors considered thus are representative of the training images where the higher vectors contain most of the information. To account for generative information, a subset of the highest eigenvectors M' (where $M' < M$), is used. The cardinality of M' is heuristically chosen. The eigenvectors are collectively referred to as the "face space" [39].

To detect faces, an image region Γ of the same dimensions as the training image is mapped to the face space by first calculating $\Gamma - \psi$ and considering the dimensions in M' only. If a region contains a face, the representation of the same in the "face space" will be small as we calculate the difference with the average face, ψ, in $\Gamma - \psi$. Similarly, the "face space" representation for a non-face will be comparatively larger. A threshold can thus determine if the region is a face. An iterative procedure over all possible locations will return a list of image patches regarded to contain faces by the algorithm. The input image can be scaled to different factors and the procedure repeated to detect faces of all sizes. Figure 3.5 shows an average face ψ and a set of eigenfaces.

Sung and Poggio [42] introduce a distribution-based face detection approach. Canonical face and non-face patterns are collected as training images from which representational clusters are listed. The clusters are approximated to a number of multidimensional Gaussians. The clusters are further refined by using non-face clusters, also approximated as Gaussians, that surround the face clusters. Distance measures between the collection of clusters detect faces in sampled regions of the input image.

A discussion on appearance based models would be incomplete without a mention of the neural network based face detection system by Rowley et al. [28, 43]. An ensemble of neural networks trained on preprocessed images is used. Training images are 20×20 pixels in dimension and are histogram equalized[3] so as to render the system illumination invariant. To deal with the false positives, a number of arbitration methods such as an attentional neural network to merge detection results from different neural networks are employed. If faces are falsely detected in images where there are none, the regions assumed to contain faces are added to the non-face training image collection so as to prevent such detections in the future. The end result is a very robust face detection framework. To conclude, we note that appearance-based methods remain to offer one of the most competitive results to-date.

3.2.5 Sampling-based Approach

Sampling-based approaches probe local visual descriptors to learn facial characteristics. In contrast to some of the previous approaches, the descriptors are estimated directly from the training data and do not typically conform to any template.

[3] Histogram equalization is an image processing technique to modify the dynamic range of image intensities and improve contrasts [44, 45].

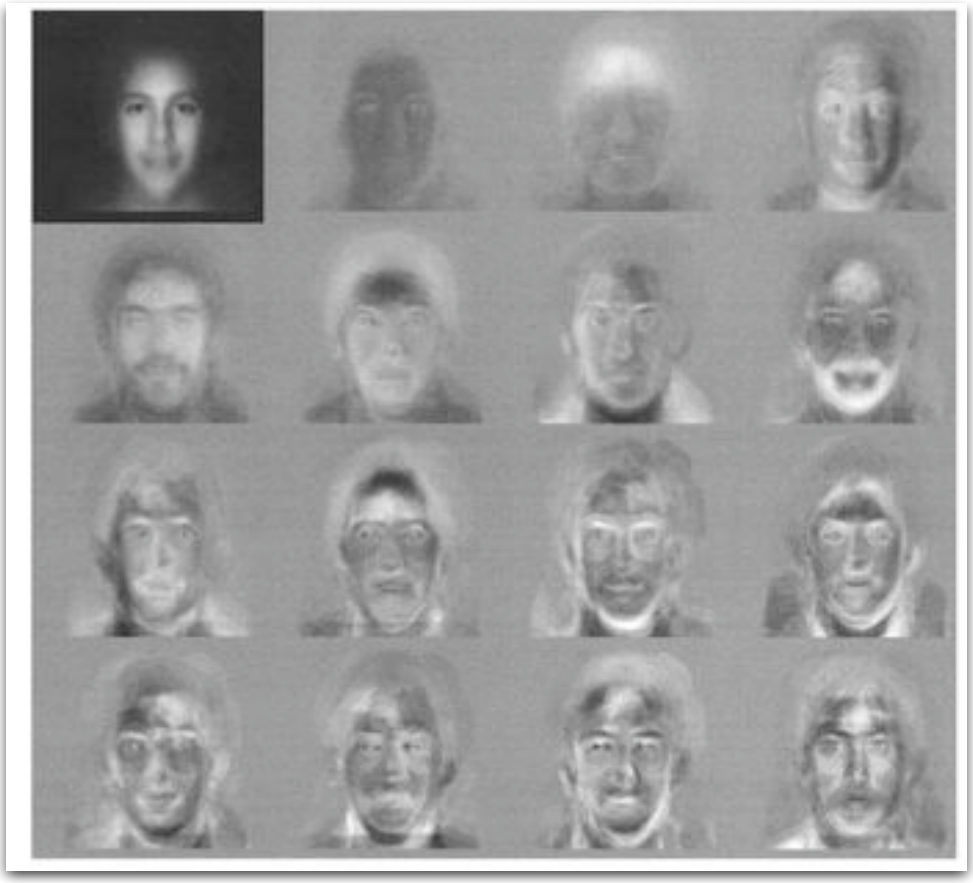

Figure 3.5: An average face followed by a set of eigenface images [41].

The task of finding the descriptors is implemented by sampling the training images at regions of interest. Approaches can sample images densely [23, 46], randomly [10] or at salient points [47]. An exhaustive procedure is to sample all possible pixel positions but this is computationally prohibitive and hence avoided. Figure 3.6 shows the sampling of a few rectangular regions in training images.

Classifiers are built by first quantizing the descriptors using a set of filters to obtain features. The features are thresholded as decision criteria. Rectangular regions of the same dimension as the training images are sampled iteratively in the input image. Features at equivalent locations are extracted and compared with those collected by the classifier. Distance measures such as euclidian or mahalanobis are used to nominate the class label (face or non-face). It follows that this approach is *generic*: as face-specific properties are generally not modeled as part of the algorithm, it can also be used to

Figure 3.6: Sampling rectangular regions for training [48].

detect objects other than faces.

A renowned [29] sampling-based learning framework was proposed by Viola and Jones [30] that uses boosting to learn discriminant features[4]. This approach builds many binary decision stages. Each stage focuses on features that leads to the minimal error on dividing the training set into two groups. Since there is usually never a single feature that returns zero error [30], the learning algorithm builds attentional stages on features that return the minimal error for each additional stage. At the end of the training procedure we obtain a cascade of simple classifiers that have learnt a small set of criteria to effectively determine if a region of the input image contains a face or not. As it is the subject of all our performance evaluations, we briefly review a summary of the approach.

For a two-class problem, Equation 2.1 can be written as $\{(\Gamma_n, c_n) : n = 1, \ldots, N\}$, where $c_n \in \{face, non\text{-}face\}$ is the corresponding class label. At each stage t, the approach computes a classifier that divides the training data at this stage into two with the smallest error. The error with respect to a classifier h_j is evaluated as,

$$\varepsilon_j = \sum_n w_n |h_j(\Gamma_n) - c_n| \tag{3.2}$$

where $h_j(\Gamma_n)$ is 1 if the feature in Γ_n exceeds the threshold and 0 if not. Equation 3.2 thus computes the number of training images that will be misclassified on this classifier. All possible classifiers are considered and the classifier with the lowest ε is chosen as h_t for the stage t.

Initially, the weight w_n is set to the same value for every image in Γ_1^N. In subsequent stages, the weight is influenced by the error calculated in Equation 3.2 for the previous stage as,

$$w_{t+1,n} = w_{t,n}\beta_t^{1-e_n}$$

[4]Discriminant features of a class are features specific to the class, i.e. features that distinguish this class from the others.

$$\beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t} \tag{3.3}$$

where, e_n is 0 if Γ_n was correctly classified by the classifier and 1 if not. The ε_t represents the error with respect to h_t.

After T stages, the classifier is given as,

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise,} \end{cases} \tag{3.4}$$

where $\alpha_t = 1/\beta_t$. Subwindows of the same dimension as the training images are sampled from the input image and if the classifier returns 1, then the same is nominated to be a detected face. The sampling continues at different scales to return all the faces in the input image.

3.3 Discussion

We observe that there is clearly a noticeable overlap between the approaches. For instance, edge maps and elliptical structure templates (Sirohey [33]) is also a template based approach although it is presented as a feature-based approach. Some approaches can thus be classified into two categories. Further, the boundary between knowledge-based approaches and template-based is also fuzzy [25]. Template-based approaches also model the human intuition on the morphological characteristics of a human face.

We note that earlier face detection approaches suffered from the inability to detect faces of different scales and side profiles. Detecting differently sized faces in the same input image have since been addressed by many [28, 43, 30, 42, 25].

Typically, to detect faces in the input image, a detection framework uses the sliding subwindow approach. Image patches (subwindows) are sampled, iteratively, from the left to right and top to bottom. The dimension of the subwindow is equal to the dimension of faces in the training images. Effectively, the sliding window finds subwindows similar to the images that the framework was trained on. The face detector returns a face or a non-face label for the sampled subwindow. Similarly, the iterative sampling routine lists all subwindows nominated to contain faces. To account for faces of sizes different from the dimensions of the training data, the input image is both upscaled and downscaled. The scaled input images are now searched for face subwindows. A post-processing step merges the detections to return the location and extent of all the faces in the input image.

To address side profiles, Rowley et al. [43] institute a "router" network that investigates possible orientations of the face to constitute a rotation invariant neural network face detector. Similar arbitration schemes are required to detect side profiles in other learning frameworks.

In the next chapter, we discuss feature extraction methods and approaches to face recognition. Feature extraction methods are either an implicit component of face detection (as in the case of feature-based approaches) or they serve as an intermediate stage to face recognition.

4 Face Recognition

Many face recognition methods extend the framework of face detection approaches to learn characteristics that differentiate one identity from another. Face recognition is applied either for the task of *identification* or *verification*. Identification is to match a given face image with a stored collection of identities and return an identity. Verification is to check the identity alleged by the given face image using a stored collection of images for this identity. Thus, in the case of identification, a similar face is searched for in the collection and the corresponding label is returned. In the case of verification, the given face image is compared with the faces assigned to the claimed identity and the status of this comparison is returned.

Prior to the recognition of faces, it is necessary to extract facial characteristics or features. Feature extraction is the task of reducing the high dimensional training data to a set of *features* to investigate properties (morphological, geometric etc.) of the data [49, 45]. We reintroduce Figure 3.1 here to reiterate the stages of a face processing system discussed in this chapter.

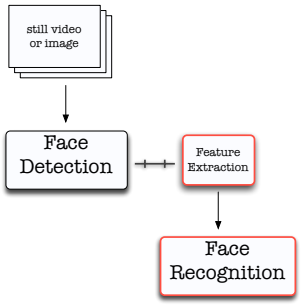

Figure 4.1: The three stages of a face processing system.

4.1 Feature Extraction

Feature extraction is the task of reducing the high dimensional training data to a set of *features* to investigate characteristics (morphological, geometric etc.) of the data [49, 45]. We have, in the previous chapter, reviewed feature extraction steps such as edge and blob detection, PCA, locating facial features etc. as an implicit component of the

face detection approaches. Similar features are also used by recognition approaches to differentiate between faces of different identities. We note that depending on the recognition approach and its respective requirements, there might be a separate feature extraction stage in addition to the one in detection.

Features are also extracted from local visual descriptors. We reiterate that local visual descriptors are representative of local characteristics of images. In Equation 2.2, we gave a general definition of a filter as:

$$f(\Gamma) : \Gamma \mapsto \Re^{M'}$$

where Γ is the input image and $\Re$ is the space of real numbers. The value of M' is dependent on the filter and is either a scalar (feature) or a vector (feature vector). We review two filters here. The first of which is the Haar filter and quantizes an image region to a scalar and the second is the Gabor filter that quantizes an image region to a vector.

The Haar filter, reminiscent of the Haar basis function [50], quantizes image regions by adding up the pixel values in that region. A simple *two-rectangle* feature calculates the difference between the sum of pixels of two regions as shown in Figure 4.2. A three-rectangle feature divides the region into three rectangular sections and calculates the difference between the sum of pixels of two of the divisions and the third. More features are illustrated in Figure 4.2.

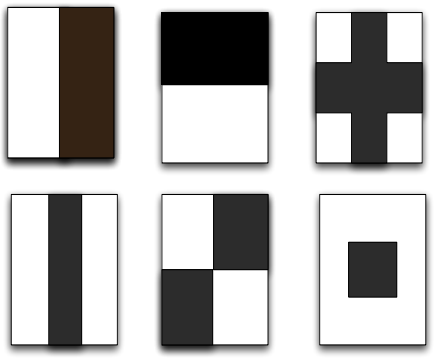

Figure 4.2: Six different Haar filters: the sum of the pixel intensities in the black region are subtracted from the sum of the pixels in the white region to obtain the corresponding Haar feature (see text).

A more complex filter is the Gabor filter [51]. Gabor filters capture localization, orientation and spatial frequencies. Gabor filters used by Yang et al. [48] are defined as

$$g_{u,v}(z) = \frac{||k_{u,v}||^2}{\sigma^2} e^{\frac{-||k_{u,v}||^2||z||^2}{2\sigma^2}} e^{\frac{-||k_{u,v}||^2||z||^2}{2\sigma^2}} \left[e^{ik_{u,v}z} - e^{\frac{-\sigma^2}{2}} \right], \tag{4.1}$$

where $z = (x, y)$, v is the scale and u is the orientation of the filter. Here, $k_{u,v}$ and σ are functions of the frequency. Yang et al. [48], for instance, consider the following definitions:

$$\sigma = 2\pi$$
$$k_{u,v} = \frac{\pi}{2}\frac{1}{\sqrt{2}^{v}}e^{i(\frac{u\pi}{8})}$$

Both u and v can take on a set of values, such as $\{0, 1, \ldots, \}$. The responses of the training data to the filter on the different u and v values serve as feature vectors. A few responses to the Gabor filter are illustrated in Figure 3.6.

4.2 Recognition from Intensity Images

Face recognition is a multi-class problem. If we consider P human identities, then we have a P class problem. Accordingly, the training data is organized as P collections. A face recognition approach, typically, is modeled either as a one-on-one problem or a one-on-all problem. In the former, each identity is paired against another and the problem is reduced to finding the likelihood of an input face belonging to the first identity over the second. Here, $P(P-1)/2$ classifiers are built. In a one-on-all problem, each identity is paired against the remaining $P-1$ identities and thus, P classifiers are built. In both of the above, extracted features are either in the *intrapersonal* or the *interpersonal* space of an identity Ω_p.

Face recognition is challenging as face images of the same person can have perceptible disparity, sometimes greater than between face images of different people due to illumination and pose variations. Further, facial structures such as beard and mustaches make recognition difficult. Approaches to face recognition thus attempt to correctly learn features from both the intrapersonal and interpersonal space for every identity using the P class training data. In this section, we review geometric approaches, a sampling based approach and two general pattern recognition approaches to face recognition.

Turk and Pentland [39] present a recognition approach that extends the eigenspace framework [41, 52, 53, 54, 39, 40, 55, 56, 57] previously discussed in Section 3.2.4. We recall that the eigenspace framework calculates the covariance matrix C for the face training data and finds the eigensolution to the same. Eigenvectors are then ordered in a decreasing order of the corresponding eigenvalues. A few of the highest (M') are chosen as sufficiently generative of the "face" space. An input image region is mapped to the face space by considering $\Gamma - \psi$ and comparing a value extracted from the same to a threshold. For instance, an input image region Γ is reduced to a vector of weights as $w_k = u_k^T(\Gamma - \psi)$, where u_k are unit vectors (and u_k^T is the transpose) and $k = 1, \ldots, M'$ are the M' dimensions considered. We thus obtain a vector $v^T = w_1, \ldots, w_{M'}$. In

the eigenspace framework for face detection, if the ν^T was below a threshold, we considered the image region to be a face.

The framework for recognition is similar. Here, Γ is an input face and ν^T is calculated as above. The average face for each identity is calculated as ψ_p and they are mapped to a vector as $u_k^T(\psi_p - \psi)$. Like above, ν_p^T is calculated for $p = \{1, \ldots, P\}$. From ν and ν_p, Turk and Pentland [39] estimate the euclidean distance measure, $||\nu - \nu_p||$. This distance measures the similarity of an input face to the identity Ω_p in the "face" space. The Ω_p with respect to which the euclidean measure ($||\nu - \nu_p||$) is the lowest is considered to be the *recognized* identity.

To account for the possibility of the face belonging to an unknown identity, the euclidean distance is compared with a threshold. If the value is greater than the threshold, the face is awarded as an unknown. In conclusion to their work, Turk and Pentland [39] list four possible scenarios that describe the relations between face space and recognition in terms of the eigenvectors for an input image Γ:

- Eigenvectors are in the face space and close to ν_p (identity Ω_p): the input image is labeled with p.
- Eigenvectors are in the face space but not close to any ν_p: the input image is a face that belongs to an unknown identity.
- Eigenvectors are not in the face space but close to ν_p: the input image is not a face.
- Eigenvectors are not in the face space and not close to any ν_p: the input image is not a face.

Moghaddam and Pentland [52], however, argue that simplistic Euclidian measures are insufficient to capture truly discriminate information. They present a probabilistic similarity measure on the intrapersonal and interpersonal space of an identity Ω_p. Image intensity difference is calculated as $\Delta = \Gamma_1 - \Gamma_2$. Intrapersonal variations, $\mathbf{\Omega}_I$, are equated to probability distributions drawn from variations in face images of Ω_p. Similarly, interpersonal variations, $\mathbf{\Omega}_E$, are drawn from variations in faces between Ω_p and the other identities.

The similarity between Γ and a representative face for Ω_p, I, is,

$$S(\Gamma, I) = P(\Delta \in \mathbf{\Omega}_I) = P(\mathbf{\Omega}_I|\Delta) =$$

$$\frac{P(\Delta|\mathbf{\Omega}_I) \cdot P(\mathbf{\Omega}_I)}{P(\Delta|\mathbf{\Omega}_I) \cdot P(\mathbf{\Omega}_I) + P(\Delta|\mathbf{\Omega}_E) \cdot P(\mathbf{\Omega}_E)}$$

where $P(\Delta \in \mathbf{\Omega}_I)$ is the Bayes maximum a posteriori. The prior probabilities are estimated from dimensionality reduction methods, such as PCA.

Fisher Discriminant Analysis (usually called Linear Discriminant Analysis or LDA) [40, 26], another geometric approach, is drawn from the observation that the traditional eigenface solution calculates the scatter matrix[1] of the complete set of face training images and does not effectively discard the intrapersonal variations. Essentially, the eigensolution of this scatter matrix includes between-class (interpersonal) scatter and within-class (intrapersonal) scatter as well. LDA, thus, optimizes on an eigensolution that highlights this difference. We rewrite eigenfaces in terms of the scatter matrix as:

$$W_{opt} = \arg\max_{W} |W^T S W| \tag{4.2}$$

where S is the scatter matrix of the face training data. W corresponds to w_k and W_{opt} corresponds to the M' dimensions chosen. LDA, in contrast to the eigenface framework, considers two separate scatter matrices, S_w and S_b referred to as the within and between class scatter matrix. They are calculated as,

$$S_w = \sum_{p=1}^{P} Pr(w_p) C_p$$

$$S_b = \sum_{p=1}^{P} Pr(w_p)(\psi_p - \psi)(\psi_p - \psi)^T$$

where C_p denotes the covariance matrix of the class p. The probability $Pr(w_p)$ is a prior probability of a class and is usually $1/P$ in practice [26]. To optimize on a set of eigenvectors that define the between class matrix and within class matrix, an operator such as a determinant is considered. Equation 4.2 is modified to,

$$W_{opt} = \arg\max_{W} \frac{|W^T S_B W|}{|W^T S_W W|} \tag{4.3}$$

where W_{opt} are the eigenvectors to be considered. Once the eigenvectors are extracted, any similarity measure such as the euclidean distance ($||\nu - \nu_p||$) can be used for recognition.

A sampling-based approach to recognition was proposed by Jones and Viola [58] that extends the adaptive boosting framework for face detection [30]. They model the recognition problem as a comparison between two faces, Γ and I represented as $F(\Gamma, I)$. This is referred to as the *face similarity* function and is defined as,

$$F(\Gamma, I) = \sum f(\Gamma, I)$$

[1] We had previously used the covariance matrix. With respect to Equation 3.1, the covariance matrix and the scatter matrix are related as $S = NC$.

where $F(\Gamma, I) \in \Re$. As above, Γ is the input face and I denotes a face of an identity with respect to which the recognition is carried out. Here, $f(\Gamma, I)$ is a feature that satisfies the following constrains,

$$f(\Gamma, I) = \begin{cases} \alpha & \text{if } |\phi(\Gamma) - \phi(I)| > t \\ \beta & \text{otherwise} \end{cases}$$

where ϕ is a quantization function such as the Haar filter, t is a related threshold, α and β are values which are updated on every stage depending on the number of misclassified samples, similar to Equation 3.3. The selection of features is accomplished using a slightly modified boosting model as was presented in Section 3.2.5. Like the detection approach, stages of binary classifiers are grown, each of which concentrates on achieving the minimal error defined as a function of the face images recognized incorrectly by the stage. This is analogous to the error defined in Equation 3.2. The final classifier takes the form,

$$F(\Gamma) = \text{sign}\left(\sum_{t=1}^{T} f_t(\Gamma)\right)$$

where f_t is the classifier chosen for stage t. We reiterate that the classifier is specific to an identity and the training is with respect to the intrapersonal and interpersonal spaces of an identity.

Lastly, we review face recognition with support vector machines (SVM) and Log-linear models. Both approaches consider the pixel values of the training images directly as a M dimensional feature vector. SVM are general classifiers for pattern recognition tasks [59, 60]. Typically, SVM find a linear classifier, a separating hyperplane, which divides a two class problem with the maximum generalization. Consider a two class training data (Equation 2.1) as $D = \{(\Gamma_n, c_p) : n = 1, \ldots, N; c = +1, -1\}$. A hyperplane, defined as $\mathbf{w} \cdot \mathbf{\Gamma} + b = 0$, is called the optimal separating hyperplane if it separates the training data with zero error and the distance between the data and the hyperplane is maximum [61].

Figure 4.3 depicts the optimal separating hyperplane. The training data elements that belong to class +1 are represented as circles and the data elements that belong to class -1 are represented as triangles. There are a number of candidate separating hyperplanes, two of which are drawn in the figure. However, the one that is maximally distant to both of the classes is chosen as the optimal separating hyperplane. The data elements closest to this hyperplane are called the support vectors. The distance between the support vectors and the hyperplane[2] is $1/||\mathbf{w}||$. SVM theorize that such a hyperplane offers the highest possible generalization.

[2]Calculated as the distance of a point to the plane $\mathbf{w} \cdot \mathbf{\Gamma} + b = 0$.

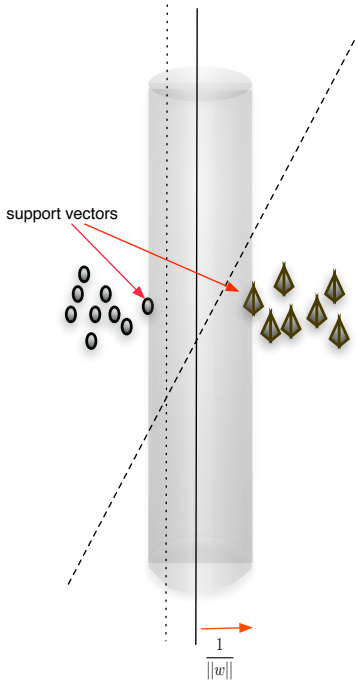

Figure 4.3: A separating hyperplane found by SVM. The dotted lines are candidate hyperplanes. However, the filled line is maximally distant from both the circles and the triangles. Accordingly the elements closest to the hyperplane are called support vectors. The distance between a support vector and a hyperplane is equated to $1/||\mathbf{w}||$.

A support vector machine must satisfy,

$$c_n|(\mathbf{w} \cdot \Gamma_n) + b| \geq 1, \qquad n = 1, \ldots, N$$

where the margin between the hyperplane and the data $(1/||\mathbf{w}||)$ must be maximized. However, due to the possibility that there is no hyperplane that perfectly separates the two class problem, soft margins are allowed with a penalty function C. Support vector machines are modeled as the following optimization problem [62, 63],

$$\min_{\mathbf{w},b,\xi} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{n=1}^{N}\xi_n \tag{4.4}$$

subject to,

$$c_n(\mathbf{w}^T\phi(\Gamma_n) + b) \geq 1 - \xi_n,$$
$$\xi_n \geq 0$$

where ξ is the error on classification. It should be noted that C is the penalty and is unrelated to c, the class label. To allow for non-linear classification, kernel functions

are introduced. For instance, the radial basis function on our task is defined as

$$K(w_i, \Gamma_j) = \exp(-\gamma||w_i - \Gamma_j||^2),$$

where γ should be greater than 0. We arrive at suitable values for C and γ using cross-validation. Support vector machines are used for face recognition using the one-on-one approach for multi-class data. A total of $P(P-1)/2$ SVM are built between every possible pair of identities. The image is classified with every SVM, the returned identity is collected for each SVM and a simple voting procedure awards the final identity. For instance, if there are three identities $\{0, 1, 2\}$, there are three SVM to build: $\Omega_0|\Omega_1$, $\Omega_0|\Omega_2$ and $\Omega_1|\Omega_2$. If the SVM results with respect to an input face Γ are $\{0, 0, 1\}$ respectively, the identity awarded is 0.

Log-linear models are also general pattern recognition classifiers and calculate the Bayesian maximum a posteriori probability of an input face Γ for every identity. The scores returned are ranked to return the final identity. The probability of Γ belonging to an identity Ω_m is calculated as

$$p(m|\Gamma) = \frac{\exp[\alpha + \lambda_m f(\Gamma)]}{\sum_{m=1}^{P} \exp[\alpha + \lambda_m f(\Gamma)]}, \tag{4.5}$$

where f is a feature of the input face and the α accounts for bias in the model. The feature, as we mentioned above, are the pixel intensities as an M dimensional vector. The sum in the denominator in Equation 4.5 is referred to as the normalization constant. The λ_m is calculated from the corresponding training data using the generalized iterative scaling algorithm [64]. The generalized iterative scaling algorithm converges on the distribution of pixel intensities for each class. The score of the input face with respect to every identity is calculated and ranked to return that with the highest value as the final identity.

4.3 Databases

We present a brief introduction to a few important training face databases. We also present important test collections to measure the detection accuracy. Subsequently, we introduce the FERET evaluation, which is an important evaluation benchmark for recognition algorithms.

4.3.1 Training Collections

Most training data collections are common to both detection and recognition. The individual approach of detection, feature extraction or recognition may have additional constraints on the format of the training images. For instance, an approach that relies heavily on the eye feature localization may expect training images where eye contours are known. For the sake of brevity, we do not present any annotations that might be available.

- **Face Database, Universidad Politecnica de Valencia** consists of 40000 face images [65] with minor illumination variations. The faces are perfectly cropped from the forehead to the lower jaw.
- **MIT Database** consists of the faces of 16 people. There are 27 pictures available per person taken under different conditions of illumination, scale and pose [66].

Figure 4.4: The Yale Face Database [67].

- **Yale Face Database** consists of the faces of 15 people. There are 11 available pictures per person taken under different illuminations, with or without glasses and expressions [67]. A sample set of images are shown in Figure 4.4.

Figure 4.5: The Yale Face Database B [68].

- **Yale Face Database B** consists of the faces of 10 people. They are taken under 576 different viewing conditions [68]. A sample set of images are shown in Figure 4.5.
- **PIE Database** consists of the faces of 68 people. They are taken under 60 different conditions [69].

- **Cohn-Kanada AU Coded Facial Expressions** consists of the faces of 100 people. They are taken with 23 different facial expressions [70].

- **AT & T Face Database** consists of the faces of 40 people. There are 10 available pictures per person taken under different illuminations, with or without glasses and different facial expressions [71].

Figure 4.6: The BioID Face Database [72].

- **The BioID Face Database** consists of the faces of 23 people. There are a total of 152 high resolution images [72]. A sample set of images is shown in Figure 4.6.

Figure 4.7: The KBSG Face Database.

- **KBSG Face Database** consists of approximately 30 faces with background per person of 4 people. The images were taken under typical conditions, with respect to illumination and background, that RWTH-Aachen RoboCup@Home robots [73] will encounter. A sample set of images is shown in Figure 4.7. The images include variations in pose, illumination and multiple faces in the same background. We note that the authors created the database and the images were captured by the aforementioned robot.

4.3.2 Test Data Collection

One of the earliest test collections was by Sung and Poggio [42]. The second collection created by Sung and Poggio consisted of 23 low resolution images with complex backgrounds and faces occupying relatively smaller areas.

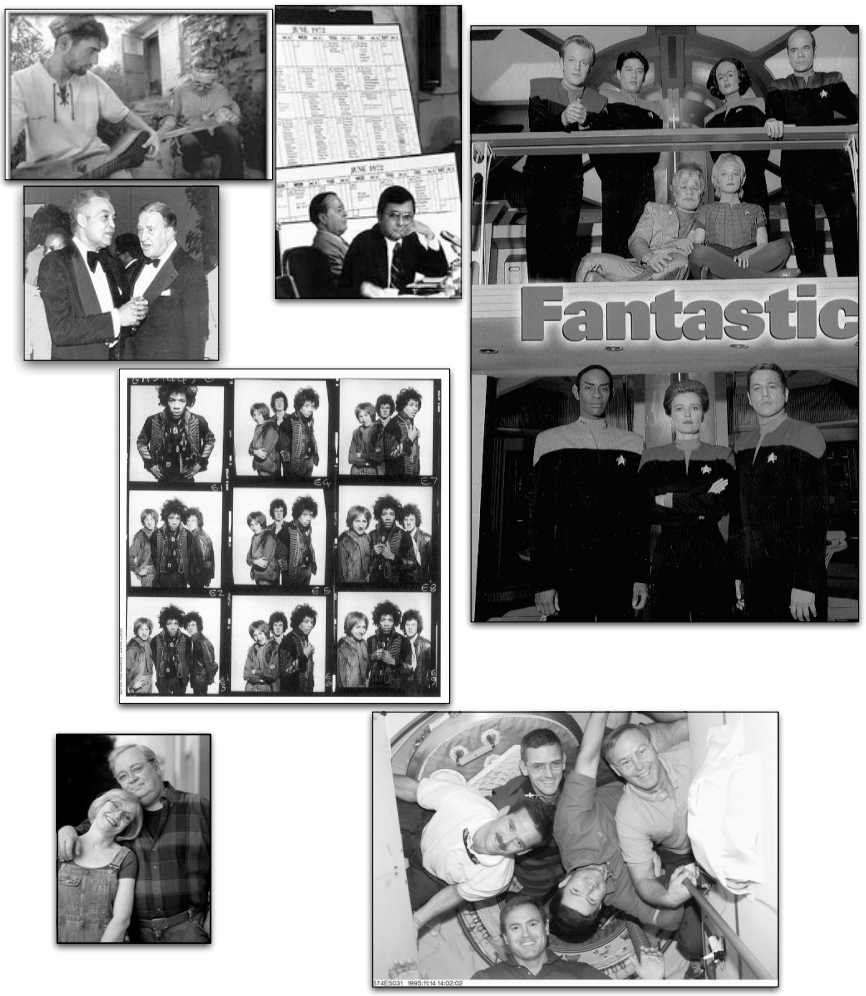

Figure 4.8: A few samples from the CMU test data collection. In addition to frontal pose face images, the collection includes rotated faces, complex backgrounds and multiple faces, in close proximity of each other.

Images of comparable complexity are also offered as part of the CMU test collection [74]. The CMU collection has 130 images with 507 frontal face images. The collection also has a few rotated faces which makes the detection task more challenging. Another test collection, the CMU Profile Face Images [75] contain 50 images with 223 face

profiles (most at an angle) also in complex backgrounds. The above collections can be considered to be real-life scenarios as they created from an variety of sources including music album covers, family and celebrity pictures, stills from TV shows etc. Figure 4.8 shows a few samples.

4.3.3 FERET Protocol

The FERET protocol is a set of evaluations and it remains to be the most important data collection for face recognition. The FERET protocol is intended to be a framework that presents a real-world setting for recognition systems [76, 26].

Figure 4.9: Sample images from the FERET Protocol [76].

Figure 4.9 displays a sample set. The images provided by FERET are available free of charge. The evaluations measure identification accuracy, false alarms and recognition rates with pose variations. The images are annotated with the conditions under which the photographs were taken which include different illumination and pose conditions. The database also includes images captured under the same conditions but on different dates to serve as probe images for recognition evaluations.

5 FRAMEWORK

In this chapter we detail our framework. Our proposal was towards a unified framework for face detection, recognition and learning of human faces with random forests.

In Section 5.1, we present the theoretical foundations of random forests. In Section 5.2, we review training data requirements of a learning framework, especially that of a face processing system. In Section 5.4, we describe the procedure to build random trees for our framework. We present face detection with random forests in Section 5.5. We present our arbitration techniques in Section 5.6. We finally present face recognition and face learning with random forests in Section 5.7 and Section 5.8.

5.1 Theoretical Foundations of Random Forests

In Section 1.2, we mentioned that random forests have distinct advantages over classical decision trees. In this section, we clarify our claim and present the theoretical foundations of random forests.

5.1.1 Decision Trees

Classic Decision Trees [18], such as C4.5 and its earlier cousin ID3, recursively grow trees top down by maximizing the information gain at the nodes [77, 78] with respect to the training data. Training data is typically of the form,

$$D = \left\{ \left(\Gamma_n, c_p \right) : n = 1, \ldots, N; p = 1, \ldots, P \right\} \tag{5.1}$$

where, Γ_1^N are the data elements and c_1^P are the corresponding class labels. Further, the data elements, Γ_n, are M dimensional i.e. they have M attributes. C4.5 (and ID3) search for the attribute that offers maximal information gain at the *parent* node. This search is geometrically a hyperplane that best divides the training data and characterized by any of the M attributes taking on a specific value. If we model an attribute to be a discrete random variable, X, the information gain of this hyperplane is given with the Shannon Entropy as

$$H(X) = -\sum_{m=1}^{P} p(x_m) \cdot \log p(x_m), \tag{5.2}$$

where $p(x_m)$ is the probability of the division with respect to class m. Thus, the algorithm finds the hyperplane, parallel to an axis in the M dimension, with the lowest H and divides the training data into two groups. The groups serve as the training data to the *child* nodes. The procedure recurs at the child node. Algorithm 1 briefly summarizes our review.

Algorithm 1 A decision tree

Require: training data collection, D

1: **for all** attributes **do**
2: calculate information gain with this attribute as the hyperplane
3: **end for**
4: choose attribute with highest information gain
5: create a *decision* node and divide the training collection
6: recur procedure till nodes encounters no more examples

As Russell and Norvig [18, 21] discuss, a serious limitation of decision trees is that of *overfitting*. Overfitting is the condition of a grown decision tree that is meticulously consistent with the training data and yet not learnt "useful" information. The tree has, in this case, maximized information gain on non-critical attributes due to the high dimensionality of the training data and the attributes are too general to be of any use. The shortcoming is critical to our work since image data is very high dimensional. An unseen example with a few variations will be incorrectly classified. The standard technique to deal with overfitting is with *decision tree pruning*. Decision tree pruning seeks to reduce the redundancy of the decision nodes. It traverses the tree to find nodes with zero information gain and shortens the tree by removing such nodes. However, decision tree pruning equivalently reduces the classification accuracy of the decision tree.

5.1.2 Random Forests

It remains that decision tree classifiers are popular due to their intuitive appeal and easy training procedures. Pruning procedures, as described above, might increase generalization accuracy but simultaneously reduce the classification accuracy on the training samples. There is no classical decision tree approach to increase both classification and generalization accuracy [21]. Tin Ho [21] discusses that this limitation lies not in trees themselves and introduces *oblique trees*. Oblique trees do not seek hyperplanes parallel to any axis, as was with classical decision trees. Instead, the two farthest data

samples are estimated with Euclidian distance measures and connected to each other. A hyperplane perpendicular to the projection of this connection serves as the division criteria.

While such hyperplanes avoid redundant attributes, the more important generalization accuracy does not necessarily improve. To address generalization we need a collection of classifiers, each of which should be trained independently [22, 21, 1].

Subsequently, Tin Ho introduces a *random forest*. "Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all the trees in the forest" [1]. A random tree is built by choosing components from random subspaces [21, 22]. Each random tree is trained on the entire training data. However, the trees randomize feature selection and decision criteria. Since we assume the training data to be M dimensional, there are 2^M random subspaces to choose from. Ho further discusses that very high classifier accuracies can be achieved with an increasing number of trees.

The underlying nature of random forests is that of building classifiers independently. This makes their construction inherently parallel, allowing us the flexibility to exploit parallel computing architectures. It is, as a result, referred to as a *parallel learning algorithm* [22].

We note that classifiers from random subspaces are not the *only* complex classifiers possible. Groups of classifiers built with cross-validation, bagging and boosting are all relevant multiple classifiers. Cross-validation is another approach to address overfitting [18]. Here, the training data is divided into sets of two collections with the learning framework training on one and an intermediate test procedure measures generalization errors on the other. Further, while bagging [19, 78] allows training of different classifiers using overlapping and randomized subsets of the training data, boosting [21, 79, 22] builds stages of classifiers with each stage concentrating on the samples misclassified with the previous stage. We have previously, in Section 3.2.5 and Section 4.2, presented approaches that use boosting to detect faces and recognize faces respectively.

We summarize and note that random forests create a large number of independently trained classifiers by the random sampling of rectangular features. Breiman [1] presents a discussion on the convergence of random forests and why overfitting will not occur. While an exhaustive theoretical review of this work is beyond the scope of this thesis, it is interesting to note that the generalization error converges to a limit as the number of trees in the forest increases. In the next section, we review two important parameters that directly influence the generalization error of a random forest.

5.1.3 Generalization Error

In this section, we present a brief summary of the mathematical treatment on random forests by Breiman [1]. The principal objective is to obtain a relation for the generalization error of random forests. Breiman derives an equation for the generalization error in terms of the *strength* of each random tree and the *correlation* between them.

Any application of random forests is characterized by a random vector θ that samples the training data. For instance, in the approach taken by Marée et al. [5, 6, 9, 10], θ represents a random rectangle. Amit et al. [80] build classifiers with a random selection of binary geometric features. The θ, in this case, is an integer l such that $l \in \{1, \ldots, L\}$ where L is the number of geometric features considered. A random tree is a classifier defined as $h(\mathbf{x}, \theta_k)$ where $k \in \{1, \ldots, K\}$ is the index of the tree in the forest and $\mathbf{x}$ is the input vector. The value $y_{(k)} = h(\mathbf{x}, \theta_k)$ is the class label of $\mathbf{x}$ as determined by random tree k. We note that the space of input vectors is $\mathbf{X}$ and the space of class labels is Y. The random forest, a collection of classifiers, is correspondingly defined as,

$$\{h(\mathbf{x}, \theta_k); k = 1, \ldots, K\}. \tag{5.3}$$

It is important to note that each θ_k represents an independent sampling and the sampling is from the same distribution. This implies that the random features chosen for our approach are taken from the same training data for each tree and the features are independently chosen from each other. The random trees defined cast a vote each and the votes are combined to return the class label y.

Let us now consider any general classifier $h_k(\mathbf{x})$. For an ensemble of the classifiers $\{h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_k(\mathbf{x})\}$, Breiman introduces the *margin function* as:

$$\mathrm{mg}(\mathbf{X}, Y) = \bar{\mathrm{a}}_k \mathrm{I}(h_k(\mathbf{X}) = Y) - \max_{j \neq Y} \left(\bar{\mathrm{a}}_k \mathrm{I}(h_k(\mathbf{X} = j)\right) \tag{5.4}$$

where $\bar{\mathrm{a}}$ represents the average and I the indicator function. The indicator function returns 1 if the relation specified holds and 0 if not. For instance, $\mathrm{I}(h_k(\mathbf{X}) = Y)$ returns 1 in case a classifier $h_k(\mathbf{X})$ returns the correct class of the input vector. Thus, Equation 5.4 simply tells us by how much, on an average, does the classifier return the precise class label compared to the most common incorrect class label.

The generalization error is modeled using the margin function as,

$$PE^* = P_{\mathbf{X},Y}\left(\mathrm{mg}(\mathbf{X}, Y) < 0\right) \tag{5.5}$$

which is the probability over the space of the input vectors and class labels.

Breiman substitutes the definitions in the random forest framework. The margin function, now termed mr, is defined as,

$$mr(\mathbf{X}, Y) = P_\theta\left(h(\mathbf{X}, \theta) = Y\right) - \max_{j \neq Y}\left(P_\theta\left(h(\mathbf{X}, \theta) = j\right)\right) \tag{5.6}$$

where θ is the random sampling of rectangular features as defined above. It is clear that *mr* is closely related to the generalization error. Breiman thus defines the *strength* of the ensemble (Equation 5.3) as,

$$s = E_{\mathbf{X},Y}\left(mr(\mathbf{X}, Y)\right). \tag{5.7}$$

where E is the mathematical expectation. The mean correlation $\overline{\rho}$ between the classifiers is derived as a function of E, the standard deviation and the random vector θ_k. Breiman obtains the upper bound on the generalization error as:

$$PE^* \leq \overline{\rho}\frac{(1-s^2)}{s^2}. \tag{5.8}$$

The equation above gives us a clear indication that the generalization error is influenced by two factors, the correlation between the random trees and their individual strengths. Further, the mean correlation is related to the independence of θ_k and thus, the generalization error is directly proportional to the randomization and the independence of θ_k. Breiman further derives that as the number of random trees becomes large (tends to infinity), the generalization error converges to a limit. Although in practice, we do not intend to grow an infinite number of random trees, growing as few as 10 trees presents very competitive results [1, 7, 21, 22, 11, 9, 6, 10, 12, 41].

5.1.4 Conclusion

We summarize, briefly, the benefits of growing random forests with a random sampling of features. The advantages are [16, 1]:

1. **Straightforward Learning.** The random sampling of features does not impose any explicit template or model. A generic algorithm learns characteristics directly from the training data.
2. **Local Representation.** The randomly sampled features directly analyze the image region without explicitly extracting geometric primitives such as edges.
3. **Classification with Occlusion.** As discussed in Chapter 2, approaches that use random forests are able to classify and detect partially occluded objects since not all the sampled regions need to be accurately classified. The voting procedure allows for a few sampled subwindows to be falsely classified.
4. **Parallelization.** Equation 5.8 emphasizes the independent training of random trees and as a result, parallel architectures can be used to grow each tree of a random forest.
5. **Fast Training Time.** Breiman discusses the fast training time of random forests. As random features are selected, no computational time is spent on searching for the most discriminative attribute. Thus, at each node only a single decision criterion is instantiated.

6. **Comparable Accuracy.** Breiman also discusses that random forests perform favorably in comparison to adaptive boosting based approaches. This is attributed to the fact that multiple classifiers are built, each with a decent classification accuracy.
7. **Low Generalization Error**. As we discussed in Section 5.1.3, the generalization error tends to a limit as the number of random trees grown increases.
8. **Integrated approach for tasks.** Creating a collection of learned features and querying them is a part of the same framework. This factor is influential in presenting a unified approach to detection and recognition as we discuss in Chapter 5.

5.2 Training Data Requirements

A learning framework is characterized by two entities: a learning algorithm and training data. The learning framework describes a procedure for extracting meaningful information from the training data. The training data, correspondingly, must be representative enough for all the classes. To increase robustness and the capability to work in real life scenarios, annotated training data with real life backgrounds are included. Thus, the algorithm is trained to classify, detect or recognize objects in spite of the complex backgrounds encountered when deployed.

Annotations are dependent on the problem definition. Typically, prospective training data for face detection are images with faces that cover the entire image as in case of Figure 4.4 or Figure 4.5. However, faces in bigger backgrounds as in case of Figure 4.6 or faces in complex backgrounds as in Figure 4.8 can also serve as training data. Annotations, here, detail the exact location and extent of every face in the images. Further, training images for face recognition systems require that the faces are labeled with the identity of the person.

We note that for face detection or recognition systems that principally work with facial features such as eyes and noses, an additional requirement is imposed on the training data. Training images here are annotated with the position and extent of relevant features.

As described in Equation 5.1, the training data is usually of the form

$$D = \left\{ \left(\Gamma_n, c_p \right) : n = 1, \ldots, N; p = 1, \ldots, P \right\},$$

where Γ_1^N represent the N training images, c_1^P represent the P class labels. The image Γ_n is considered to be M dimensional. We further denote each pixel value with $\Gamma_n(x, y)$ which denotes the pixel intensity at the (x, y) coordinate of Γ_n.

Face detection is a two class problem: to identify if a sample is a face or not a face. The training images for the face detection, accordingly, are divided into two: face images and background images. Any image that does not contain faces can effectively constitute the background images collection. This may include domestic objects such as bikes and cars, animals, and any typical background such as walls and furniture. Typically, the size of the non-face collection exceeds the face collection in the training data for face detection. This is attributed to the fact that a larger group of entities constitute as non-faces. As low false positives is usually desired, better performance is achieved with a more comprehensive non-face collection. Face recognition, on the other hand, is modeled as a P class problem with one class assigned to each of the P identities. The training images for face recognition are, accordingly, divided into P collections.

5.3 Learning Framework

In this section we discuss our learning framework. The learning framework details the randomization of decision criteria, the filter and features used and an efficient representation for computation that we borrow from Viola and Jones [30].

5.3.1 Decision Criteria

A randomized decision criterion consists of a *randomly sampled rectangle*, i.e. of random dimensions and at a random location, and an accompanying *randomly chosen threshold*. We present an overview of our learning framework and the decision criterion.

Our learning framework is comprised of growing random trees from the labeled training data. Random trees are recursively built in a top-down manner. At each node t, the training images at the node $\mathbf{\Gamma}_t$ are split into two groups, $\mathbf{\Gamma}_l$ and $\mathbf{\Gamma}_r$. The groups correspond to the training images at the left and right branches of the node respectively [12]. The split is the result of a decision criterion acting on quantized random patches.

The split is disjoint with $\mathbf{\Gamma}_t = \mathbf{\Gamma}_l \cup \mathbf{\Gamma}_r$ and $\mathbf{\Gamma}_l \cap \mathbf{\Gamma}_r = \emptyset$. The nodes t_l and t_r correspond to the left and right nodes of t. The image collections $\mathbf{\Gamma}_l$ and $\mathbf{\Gamma}_r$ are now the training data at t_l and t_r respectively and the procedure is repeated. The tree is grown until the image collection at the node consists of images from only one class. For instance, in the case of the two class face detection problem, the training images are recursively split until the node has either only faces or only non-faces. An alternate growth stopping criterion are the number of nodes that the tree is allowed to grow. This criteria is borne out of the need to constrain memory requirements when it is known in advance or experimentally estimated that a specific number is sufficient for a desired accuracy.

The framework randomly samples the training images. Random sampling signifies that a patch whose position and extent is completely random (but within the dimensions of the training images) is considered. Needless to say, the position and extent is the same in all the training images at a particular node. However, if the training images are not all of the same dimensions, either all of them are rescaled to $K \times L$ (where $K * L = M$) or the position and extent is chosen relatively in every image with respect to an M dimensional image. The patch selected is represented as $\{x_t, y_t, h_t, w_t\}$ where x_t is the x-coordinate of the top left corner of the patch, y_t is the y-coordinate of the top left corner of the patch, h_t is the height and w_t is the width of the patch.

The framework selects a random threshold. The threshold is also randomly chosen in the range of minimal and maximal values that a quantized patch can take. Mossmann et al. [12] and Marée et al. [9] refer to random trees with thresholds also chosen randomly as *extremely random trees*. We however exclude the qualifier for the sake of brevity. We denote the threshold for the node t as θ_t. We summarize the above as a tuple, $T_t = \{x_t, y_t, h_t, w_t, \mathbf{\Gamma}_l, \mathbf{\Gamma}_r, \theta_t\}$ and refer to it as a *test*. Here, $\mathbf{\Gamma}_l$ and $\mathbf{\Gamma}_r$ are not specific to a test, but since we generally instantiate one test per node t, we consider all of them together.

We note that the threshold can also be a range as opposed to a single number. Instead of probing if the quantized scalar value exceeds θ_t, we could assert if the value lies in $(\theta_{t1}, \theta_{t2})$. However, this is a minor detail and as the thresholds are randomly chosen, the number of thresholds is not important.

5.3.2 Features

The patches sampled thus are quantized as a scalar quantity or a lower dimensional vector quantity. The purpose of quantization is to extract meaningful information and execute operations, such as a test T_t defined above. A *filter* is applied on the patch and the output of the filter is the *feature*. Earlier, in Section 4.1, we discussed Haar filters and Gabor filters. A Haar or a Gabor filter applied on a region of the image gives us the Haar feature or the Gabor feature of that region.

The desired properties of any feature is that it contains both generative and discriminative information with respect to the training data. Generative information are properties that can reconstruct the image, such as the M' dimensional vector in PCA. Discriminative information between two classes are properties that highlight the differences between the classes. Haar features are generative and discriminative [54]. In the adaptive boosting based face detector [30], Haar filters successfully learnt discriminatory features between the face and non-face class. Tao et al. [81] discuss that bases from a binary subspace can be used as generative components of images. Haar features can be considered to be binary bases [54, 30] and hence contain generative information.

The Viola and Jones [30] face detection system offers one of the best performances to date. Although they train on Haar features, they review that Haar features are not that powerful. They discuss the strengths of more complex features such as steerable filters [82]. Steerable filters are called so because they can record responses at any arbitrary orientation and thus, it is possible to "steer" the filter to a desired orientation. This property is extremely advantageous. In comparison, Haar features are usually designed to only capture horizontal, vertical and diagonal orientations as shown in Figure 4.2. However, Haar features offer computational simplicity. As we build a large number of random trees with sampled patches of different dimensions, it is convenient to use features that can be computed easily. Further, we borrow the image representation called the *integral image* introduced by Viola and Jones [30]. We describe the integral image computation in the following section. This representation allows for very fast computations of Haar features.

It is interesting to note that Gabor features are reported to work well for face recognition [83]. However, many Gabor features are redundant. Yang et al. [48] use adaptive boosting to single out the discriminant ones as mentioned in Section 4.2. Gabor features in general do not offer the computational simplicity of Haar features and hence, we only consider Haar features for our learning algorithm.

To reiterate, Haar filters are reminiscent of Haar wavelets which are ortho-binary in nature. Haar filters usually divide a patch into rectangular components, designating alternate regions to either +1 or -1. The pixels in the regions are aggregated according to the above designation. As shown in Figure 4.2, a *two-rectangle* feature considers the sum of pixels in the black region subtracted from the sum of the pixels in the white region. A *three-rectangle* feature considers the sum of pixels in the black region subtracted from the sum of pixels in the white regions. A *four-rectangle* feature considers the sum of pixels in the diagonal black regions subtracted from the sum of pixels in the diagonal white regions. Figure 4.2 depicts a few Haar filters.

5.3.3 Integral Images

Haar features can be computed quickly using the image representation introduced by Viola and Jones [30]. The integral image is a scalar for each two dimensional pixel location. If the pixel is (x, y), the integral image is calculated as

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} \Gamma_n(x, y), \tag{5.9}$$

where $ii(x, y)$ is the integral image. We note that we use a left hand coordinate system, i.e. the origin $(0, 0)$ is the top left corner of the coordinate system. The integral image is calculated iteratively in a single pass using the following recursive definition [30],

$$s(x, y) = \begin{cases} s(x, y-1) + \Gamma_n(x, y) & \text{if } y \neq 0 \\ \Gamma_n(x, y) & \text{if } y = 0 \end{cases} \tag{5.10}$$

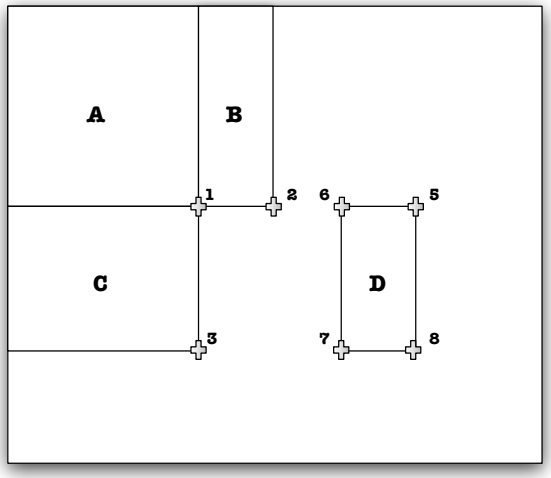

Figure 5.1: The sum of pixels in the rectangle A is the integral image at location 1. The sum of pixels in the rectangle B is $2-1$. The sum of pixels of $A+C$ is the integral image at 3. The sum of pixels in D is $8+6-7-5$.

$$ii(x,y) = \begin{cases} ii(x-1,y) + s(x,y) & \text{if } x \neq 0 \\ s(x,y) & \text{if } x = 0 \end{cases} \tag{5.11}$$

where s is the sum of pixels.

In Figure 5.1, a typical computation is illustrated. The figure denotes an integral image representation. As we see in the figure, the sum of pixels for an arbitrary region can be calculated with only four array references. This integral image calculation needs to be done only once for an image, irrespective if it is used for training or as a test image.

5.4 Growing Trees

As discussed earlier, the tree is recursively grown with $\mathbf{\Gamma}_t$ at node t using the test $T_t = \{x_t, y_t, h_t, w_t, \mathbf{\Gamma}_l, \mathbf{\Gamma}_r, \theta_t\}$. Before we review how the features are chosen, we consider the simple two-rectangle feature as shown in Figure 4.2. Here, (x_t, y_t) represents the top left corner of the sampled patch. Figure 5.2 lists the remaining coordinates.

The sum of pixels in the black region of the image patch is obtained by evaluating the values in the integral image as,

$$s(black) = (x_t + w_t, y_t + h_t) + (x_t + \frac{w_t}{2}, y_t) - (x_t + \frac{w_t}{2}, y_t + h_t) - (x_t + \frac{w_t}{2}, y_t).$$

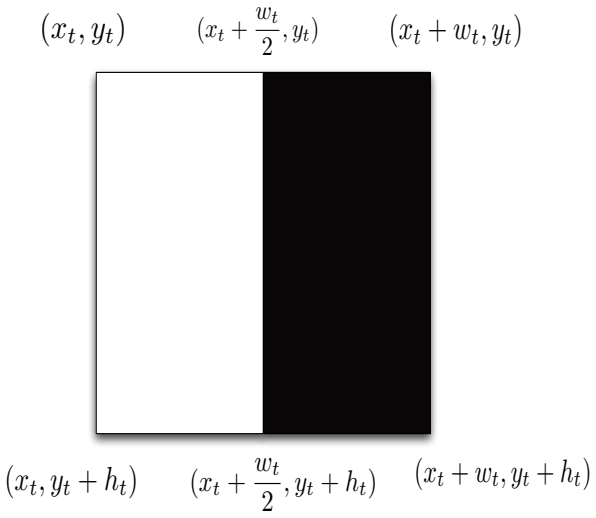

Figure 5.2: The coordinates of a two-rectangle haar feature for an image patch.

Similarly, the sum of the pixels in the white region of the image patch is obtained by evaluating the values in the integral image as,

$$s(white) = (x_t + \frac{w_t}{2}, y_t + h_t) + (x_t, y_t) - (x_t, y_t + h_t) - (x_t + \frac{w_t}{2}, y_t).$$

The Haar feature is $s(black) - s(white)$ as described previously.

As the threshold θ_t of every test is chosen in a range of numerical values, the Haar features must be appropriately mapped to this range for correct comparisons. A simple solution, for instance, is if we choose the range of thresholds to lie in $[0, 1]$. The Haar feature calculated as above can be scaled to values in $[0, 1]$ using the dimension of the rectangular region as a parameter. This process if referred to as *normalization*. The normalized Haar feature is,

$$\text{Normalized Haar Feature} = \frac{s(black) - s(white)}{h_t * w_t * 255},$$

where it is assumed that the pixel intensities are in the range $[0, 255]$.

The normalized Haar feature of every image in $\mathbf{\Gamma}_t$ is compared with θ_t. If the value exceeds the threshold, it is grouped with $\mathbf{\Gamma}_l$ and with $\mathbf{\Gamma}_r$ otherwise. The left and right branches of node t inherit $\mathbf{\Gamma}_l$ and $\mathbf{\Gamma}_r$ respectively.

Up to this point in the discussion, we did not review which of the Haar filters are used at a node t. It is evident that the learning framework must choose features that effectively divide the images at the node to act as a decision tree. As we have a set

of Haar filters, we use the Shannon Entropy to select one. The entropy of a split is calculated as,

$$H = -\sum_{P} p \cdot \log p, \tag{5.12}$$

where H is the entropy, p is the probability of a class with respect to the division and is calculated for every class in $\{1, \ldots, p\}$. In the case of the face detection problem,

$$p_f = \frac{\text{number of faces in } \mathbf{\Gamma}_l}{\text{number of faces in } \mathbf{\Gamma}_t}$$

$$p_{nf} = \frac{\text{number of non-faces in } \mathbf{\Gamma}_l}{\text{number of non-faces in } \mathbf{\Gamma}_t}$$

where $\mathbf{\Gamma}_l$ and $\mathbf{\Gamma}_t$ are as above, p_f is the probability of the face class with respect to the division and p_{nf} is the probability of the non-face class with respect to the division. For each normalized Haar feature the entropy is calculated with respect to θ_t. We choose the Haar feature with the smallest value for H. A Haar filter that proves most effective at a node is highly dependent on the training images at the node. Thus, there is usually no single Haar filter that is effective for every node of the random tree. We note that we can estimate probabilities with respect to $\mathbf{\Gamma}_r$ with an completely identical treatment.

Figure 5.3 illustrates the growth of a random tree. At the node t, there are four images, two faces and two background images. A two-rectangle Haar filter is chosen (corresponds to the region around the eye in the face images). The Haar filter is characterized by the sum of the pixels in the left component subtracted from the sum of the pixels in the right component. As the pixel values in the left and right eye is very similar, the Haar feature leads to a very small value (close to zero). Similarly, the first background image also has a homogenous appearance in the region. In comparison, the second background image has considerable disparity. On comparing the feature with threshold θ_t, the first three images are separated. The last image is placed in a leaf node and does not undergo further growth. Another two-rectangle Haar filter is randomly sampled at node $t + 1$. However, the feature does not prove discriminatory enough to divide the images at the bin. A two-rectangle filter at node $t + 2$ and the threshold θ_{t+2} separate the faces from the background image. As both children of node $t + 2$ contain images from only a single class, they remain as leaf nodes and the growth is stopped.

In the figure above, the growth is accomplished in 4 levels of the tree. In the actual execution path, this may be different. Haar features that separate image classes may be sampled after an arbitrary number of levels. Further, the threshold θ_t in the figure correctly separated similar numerical quantities. In an extremely random tree, the thresholds need not account for a separation even in the case of quantized features that are linearly separable. We can thus expect arbitrary divisions at arbitrary depths. As discussed before, the randomization affords us high generalization accuracy.

However, at the cost of generalization accuracy, the tree might be grown to large depths on redundant and non-discriminatory random patches. To circumvent this problem,

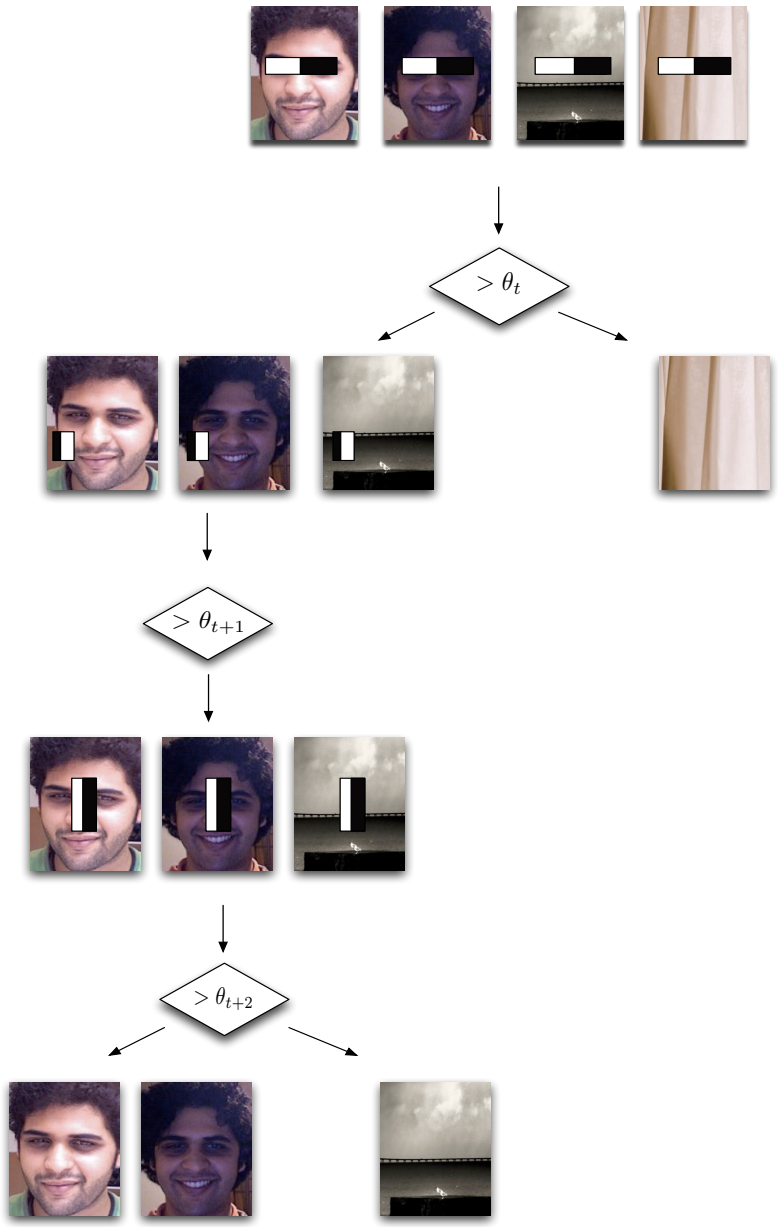

Figure 5.3: The growth of a random tree. At node t, there are 2 faces and 2 background images. A Haar feature is compared with θ_t for each image. The process is repeated until images of only one class are in the leaf node (see text).

instead of considering just a single test at a node, we consider an array of tests. In this manner, we include randomization and additionally shorten tree growth (which leads to a reduction in training and testing time). Thus, we augment the random forest

growth procedure by considering $T_t[L]$, where L is the number of tests initialized at each node. Deselaers et al. [14], for instance, set $L = 200$. We set $L = 200$ as well.

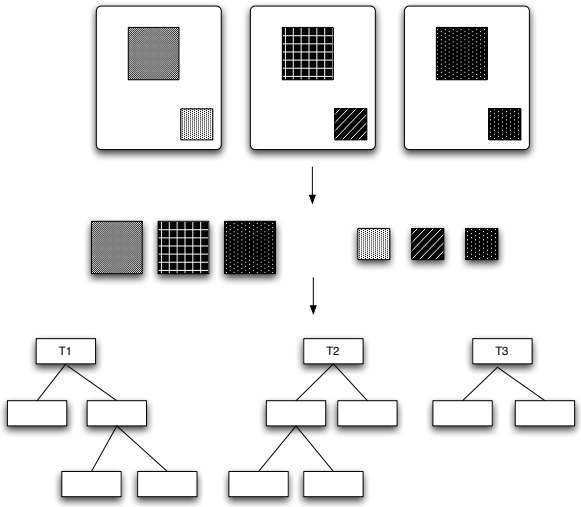

Figure 5.4: Random forest from random subsampling. Trees $T1$, $T2$ and $T3$ are grown independently.

A collection of trees is grown as outlined. Each tree can either be built using the entire training data or random subsets of the same. The trees in the forest are built independently. The forest, thus, is implicitly capable of exploiting parallel architectures. Figure 5.4 illustrates the growth of a random tree. We note that the growth is analogous to Figure 2.1. The foremost disparity is the rescaling of sampled subwindows by Marée et al. [5, 6, 9, 10].

We summarize our approach in Algorithm 2 and Algorithm 3. In the next section, we discuss the detection of faces with this framework. A tree grown with our framework to 100 nodes is presented in Figure 5.5. Every node in the tree is characterized by three fields: the node index, composition and entropy value of the best test chosen. The tree is trained on a five class training data or $P = 5$ or $c = \{0,1,2,3,4\}$. The root node is thus $\{0,(9,7,8,9,9),1.639644\}$, where the node index is 0, the number of training images for class 0 is 9, the number of training images for class 1 is 7 etc. and the entropy of the best test found is 1.639644. As the tree is grown breadth first, the last node grown is the right and bottom most leaf labeled as "node 100" in the figure. We also observe that for the above training data, 100 nodes prove almost sufficient to converge on tests that grow the tree to point of images from only one class remaining in the leaf nodes. A skeleton of the growth of the same tree is presented in Figure 5.6. The figure illustrates the top-down approach (the direction of the arrow) and the skew of the growth. The latter is dependent on the randomized decision criterion chosen at various depths of the random tree.

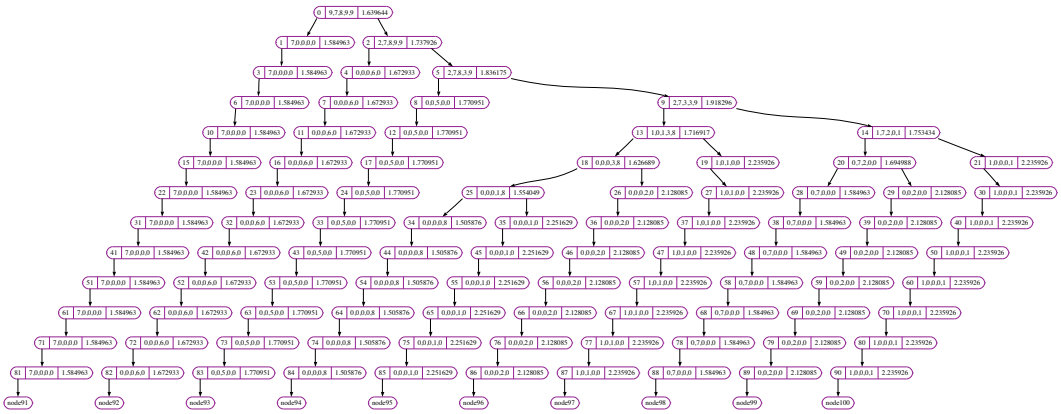

Figure 5.5: A sample tree grown with our framework to 100 nodes. Each node is characterized by three fields: node count, training data composition, entropy value (see text).

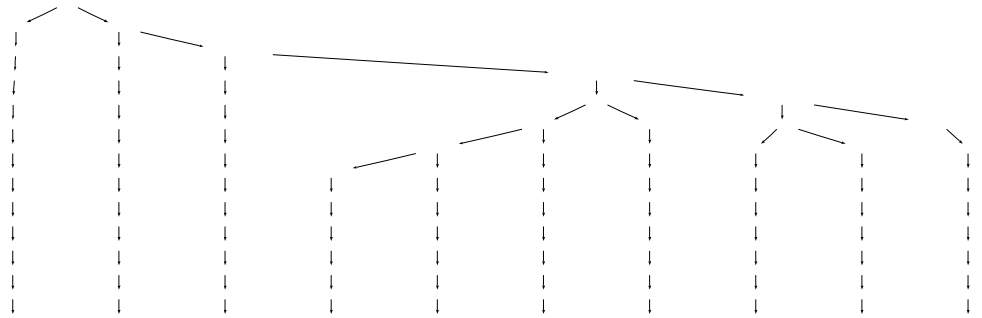

Figure 5.6: A sample skeleton of the growth of a random tree. It corresponds to the random tree in Figure 5.5.

5.5 Face Detection

The detection of human faces in an input image is achieved using a *sliding window* approach. We recapitulate that each node and the corresponding test is characterized by four training data specific parameters, x_t, y_t, h_t and w_t. Each parameter is in the range of the dimensions of the training data. In accordance, the forest learns features defined by a specification in terms of pixel size. We consider, as before, the training images to be M dimensional.

In a trivial case, the input image consists of M dimensional faces in a background. The task of face detection is reduced to returning the top left corner of each face. Let us consider a M dimensional region i'. Ideally, a random tree is grown until the leaf nodes have images from only a single class. Propagating i' down the tree is a classification

Algorithm 2 Random Tree(Γ_n)

Require: Training Images, $D = \{(\Gamma_n, c_p); n = 1, \ldots, N; p = 1, \ldots, P\}$.

let the current node be t (start with the root node)
if Γ_t contains images from only one class OR a stopping condition is reached **then**
  return t
else
  choose a test, $T_t = \{x_t, y_t, h_t, w_t, \mathbf{\Gamma}_l, \mathbf{\Gamma}_r, \theta_t\}$
  split $\mathbf{\Gamma}_t$ as $\mathbf{\Gamma}_l$ and $\mathbf{\Gamma}_r$ such that $\mathbf{\Gamma}_l \cup \mathbf{\Gamma}_r = \mathbf{\Gamma}_t$ and $\mathbf{\Gamma}_l \cap \mathbf{\Gamma}_r = \emptyset$
  let t_l = Random Tree($\mathbf{\Gamma}_l$) and t_r = Random Tree($\mathbf{\Gamma}_r$)
  let t_l and t_r be the left and right nodes of t respectively
  return t
end if

Algorithm 3 Choose a Test

Require: $|\mathbf{\Gamma}_t| > 0$.

instantiate $T_t[L]$, such that each test, $T_t[l] = \{x_t, y_t, h_t, w_t, \mathbf{\Gamma}_l, \mathbf{\Gamma}_r, \theta_t\}$
for each $T_t[l]$ **do**
  for each considered Haar feature **do**
    for each image in $\mathbf{\Gamma}_t$ **do**
      calculate feature using integral image
      normalize feature as *s(black) - s(white)*

      if $s(black) - s(white) > \theta_t$ **then**
        group the image in $\mathbf{\Gamma}_l$
      else
        group the image in $\mathbf{\Gamma}_r$
      end if
    end for
    calculate the entropy as $H = -\sum_P p \cdot \log p$
  end for
  choose the Haar feature with lowest H
end for
choose the test with lowest H
return $T_t[l]$

task. At every node, the test at that node extracts the relevant Haar feature and compares it with the threshold. As the Haar features are discriminatory, if i' is a face, then it will be separated from the majority of the non-face images down the tree. If it reaches a leaf node with only face images, i' is nominated as a face.

Propagating a region of an image on a random tree trained on face and background images returns an estimate of the likelihood that the region is a face. Thus a region classified into a leaf node with only face images is nominated as a face. A region classified into a leaf node with only background images is nominated as a background image.

However, a random tree need not be necessarily grown till the leaf nodes are composed of images from only class. Memory or time constraints are influential parameters for the learning framework. It is to be noted that the depth of tree is directly proportional to the time taken for classification as each node represents a test and a comparison. Growing extremely deep trees implicitly impacts training and classification time. Further, we even grow a collection of trees and in accordance, the growth of the tree is limited to a heuristically chosen node count or depth count.

In accordance, if a leaf node includes images from both classes, the likelihood of the node to be nominated as a face is referred to as the *Face Map Threshold* (FMP) and is defined as:

$$\text{Face Map Threshold} = \frac{\mathbf{\Gamma}_f}{\mathbf{\Gamma}_f + \mathbf{\Gamma}_{nf}} \tag{5.13}$$

where FMP is the probability of a face in the node, $\mathbf{\Gamma}_f$ is the number of faces in the leaf node and $\mathbf{\Gamma}_{nf}$ is the number of non-faces in the same. If i' is propagated to such a leaf node, a global criterion asserts if the FMP calculated is high enough to nominate i' a face.

The algorithm to detect faces in an input image is thus to iteratively sample M dimensional windows starting from the top-left corner until the bottom-right corner and propagate them on the tree. Using the FMP, we nominate if the subwindow is a face or a non-face. The iterative sampling thus examines $\sum_1^{I_w-K}\sum_1^{I_h-L}$ subwindows, where I_w and I_h represents the width and height of the input image. Algorithm 4 summarizes our review.

In the algorithm, the iterative offsets for the sampling procedure is 1, i.e. e and g proceed in steps of 1. To reduce computation time, the offsets in the X and Y direction could be increased from 1 to 2 or more. In accordance, if a window is sampled at $(0,0)$, then the subsequent window sampled is located at $(2,0)$. Once all the windows are sampled in the current row, the next window sampled is at $(0,2)$.

We have, presently, only discussed detection in the input image under the assumption that the dimensions of every face in the input image is equal to M. However, in real life scenarios, the extent of a face can take on arbitrary dimensions. Further, the same

Algorithm 4 Sliding Window for Detection

Require: Trained tree, t and an input image, i' $(K \times L)$.

for e from 1 to $I_w - K$ in steps of 1 **do**
  for g from 1 to $I_h - L$ in steps of 1 **do**
    extract window of dimensions M at (e, g)
    propagate window down the trained tree till a leaf is reached
    calculate $\dfrac{\mathbf{\Gamma}_f}{\mathbf{\Gamma}_f + \mathbf{\Gamma}_{nf}}$
    if value > global threshold **then**
      return face found at (e, g)
    end if
  end for
end for

input image can include faces of different sizes due to the distance from the camera. To accomplish detection in such images, we conduct a *multi-resolution analysis*. The input image is scaled by different factors and the algorithm outlined above is repeated at each scale. Note that the regions sampled are always M dimensional. The principal motivation for multi-resolution analysis is the observation that by downscaling and upscaling an input image, a face at some scale will be $K \times L$.

The detection process so far will result in a large number of "nominations" for the same face as every sampled subwindow a few pixels in the vicinity of a face will be nominated. Further, the multi-resolution analysis will increase both the number of nominations for a face and the extents. A good post-processing step is crucial. We review post-processing in depth in Section 5.6.

As a concluding remark, scaling the input image is not the only way to detect faces of different sizes. The sampled windows and the tests can be scaled instead. However, scaling each test in a depth first traversal of the tree is much less efficient than scaling the image just once.

5.6 Post Processing

Given the large number of prospective detected faces, arbitration is crucial for interpretable and usable results. There are three types of prospective results:

1. detections returned for a scale s by a random tree,
2. detections returned for all scales $\{1, \ldots, S\}$ by a random tree and
3. detections returned for all scales $\{1, \ldots, S\}$ by the random forest.

Our arbitration procedure considers the prospective results by post-processing detections in the same order as listed above. That is, we first arbitrate detections returned for a scale and repeat the same for each scale. Subsequently, we arbitrate detections across all possible scales to obtain a set of detections for the tree. Voting procedures merge the detections of each tree in the random forest.

A detection d is characterized by a location and the bounding box such that $d_e = \{x_e, y_e, w_e, h_e\}$, where e is the index of the detection and $e \in \{1, \ldots, E\}$. The bounding boxes from the multi-resolution analysis are downscaled or upscaled proportionately so as to be applicable to the original input image. Correspondingly, the scaled detection d'_e is

$$d'_e = \left\{ \frac{x_e}{s}, \frac{y_e}{s}, \frac{w_e}{s}, \frac{h_e}{s} \right\}, \tag{5.14}$$

where s is the scale factor.

For each scale, we collect the detections as $r_s = \{d'_1, d'_2, \ldots\}$. If at the scale of 1 three faces are detected, we have $r_1 = \{d_1, d_2, d_3\}$. If at the scale of 1.5 two faces are detected, we have $r_{1.5} = \{d'_1, d'_2\}$ where d'_1 and d'_2 are calculated using Equation 5.14 with $s = 1.5$.

The detections of the random tree are collected as $\{r_s : s \in S\}$, where S is the set of scales considered. Combining the detections is a non-trivial task as it is difficult to define a global heuristic that works with different types of input images [43].

The foremost step to the merging of detections is to define a *neighbor range*. The neighbor range of a detection window d_e is a geometric region in the vicinity of d_e such that all detection windows that overlap with the region are considered neighbors of d_e. We enumerate two strategies, also illustrated in Figure 5.7, to identify neighbors:

- Calculate the point of intersection of the diagonals of the bounding box and consider the point to be a center of a circle of diameter $2R$. The value of R can either be a fixed constant or can be related to the dimensions of the bounding box considered. All detection windows that overlap with the circle are neighbors.
- With respect to d_e, all detection windows that overlap with the rectangle are neighbors.

Following the enumeration of neighbors, the detection windows are merged. A general definition of merging is a weighted combination of all the neighbors,

$$d_{wavg} = \left\{ \frac{\sum \alpha \cdot x}{\sum \alpha}, \frac{\sum \alpha \cdot y}{\sum \alpha}, \frac{\sum \alpha \cdot w}{\sum \alpha}, \frac{\sum \alpha \cdot h}{\sum \alpha} \right\} \tag{5.15}$$

where d_{wavg} is the merged detection window. For instance, if we consider a simple average of neighbors (N_d), we can rewrite Equation 5.15 as

$$d_{avg} = \left\{ \frac{\sum x}{N_d}, \frac{\sum y}{N_d}, \frac{\sum w}{N_d}, \frac{\sum h}{N_d} \right\},$$

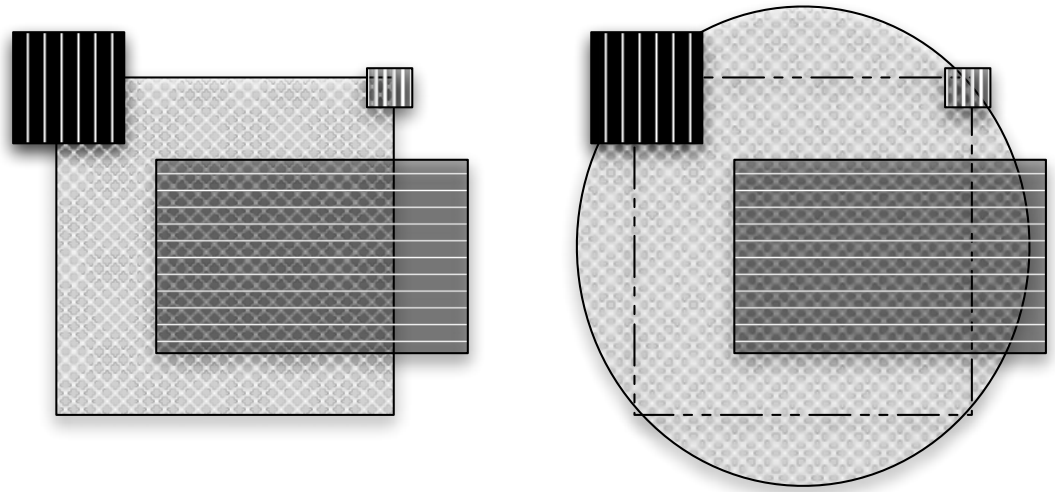

Figure 5.7: The search for neighboring detection windows by considering either all overlapping detection windows or a circular region. The neighbors are calculated with respect to the central square.

where d_{avg} is the merged window.

The value of α in Equation 5.15 can be a function of any characteristics of the detection window. For instance, the detection windows are arranged in decreasing FMP values and the detections with the lowest FMP values are discarded completely ($\alpha = 0$). The weight can also be a function of the extent of the bounding box. To exemplify, we pick the biggest bounding rectangle in the neighborhood. Although this method is simple, we show that it performs well on a broader test set in Chapter 6.

The detection windows from all the trees in the random forests are merged subsequently. We enumerate two strategies to merge,

- A detection window is nominated for the final result if and only if there is a corresponding detection window in the neighborhood in a majority of the remaining random trees. This is a weaker equivalent of the AND operator and shares similarity to simple voting scheme in classification tasks by Breiman [20, 19, 1], Marée et. al [9, 5, 6, 10] and others discussed in Chapter 2. This heuristic addresses a way to decrease the number of false positives as a detection is nominated only if the majority of trees, each with an individual generalization, appoint a region as a face.

- Detection windows are simply collected from all the random trees. They are retained as is with the notable exception of removing duplicates. This is equivalent to the OR operator. This heuristic address low detection accuracy. If a face is not detected on one tree for reasons such as overfitting, then it is probable to be detected by another random tree.

The final set of windows are returned as the list of detected human faces.

5.7 Face Recognition

As discussed in Chapter 4, the task of recognition is either that of *identification* or *verification*. Identification is the task to match a new face with a stored or trained collection of identities and return the identity of this face if it exists. Verification is the task to check if the identity alleged by the input face is correct. We model identification and verification as confidence measures based on the calculation of the *identity likelihood*. The identity likelihood is a probabilistic estimate of a face image belonging to an identity and shares similarity to the FMP discussed previously.

Our recognition framework is a classification approach. The detection phase is assumed to correctly return the location and extent of every face in the input image. The faces are scaled to $K \times L$ and propagated on a random tree in the forest[1]. To recognize the identity, the random tree is grown further till the leaf bins contain faces from only a single identity. Equivalent extensions must be carried out for all the random trees in the forest. As with detection, the identity present in the leaf node is nominated as the identity of the face.

To address the possibility that a face belongs to an unknown identity, we include an additional class in the training data. This class contains face images of many *strangers*. If the input face is propagated to a leaf node with images from this class, then the face is nominated to be *unknown*.

However, growing trees to the point where leaf nodes are composed of faces from only a single identity maybe extremely time consuming. As random regions are sampled, a discriminative Haar feature to divide face images of Ω_{p-1} from the face images of Ω_p at a leaf node may or may not be found quickly. Thus, we may introduce a constraint on the number of nodes that the tree is grown to. The number is borne out of experimental set-ups that render sufficient accuracy.

To reiterate the recognition nominations, if a scaled face image reaches a leaf node containing faces only of Ω_p, the input face is nominated that of Ω_p with the highest

[1] It is necessary to repeat the propagation of subwindows and not use any of the previous propagation results at this point as various arbitration techniques finalize the location and extent of faces from the results of detection.

confidence. The identity threshold is 1. If a scaled face image reaches a leaf node containing face images of more than one identity, then we calculate the identity threshold for this leaf as,

$$\text{identity likelihood} = \frac{\text{number of faces of the most dominant identity}}{\text{total number of face images in the bin}}$$

where the dominant identity is as estimated with a simple histogram analysis at the leaf node. If the identity threshold thus calculated exceeds a global confidence threshold, then the image is nominated as a face of the dominant identity. Identities returned from the entire forest are collected and a majority voting scheme decides the final identity to return.

We note that our recognition framework is in fact a $P + 1$ class problem. There are P identities and the additional unknown identity. A face image can be nominated as the face of an unknown identity if any of the following two conditions are satisfied:

1. The majority of the votes in the forest return the identity as that of Ω_{p+1} (stranger identity).
2. The majority margin is not large. For instance, 4 random trees nominate the face as Ω_p, 3 trees nominate the face as Ω_{p-1} and so on.

In our approach above, we presented growing every random tree further such that the leaf nodes divide the face images from different identities. A simplification of this approach is to grow another forest using only the face images as a multi-class training data. The random forest grown for face detection is not modified. The advantages of the latter approach are:

- controlling the number of nodes grown is simplified,
- propagating a scaled face down the non-leaf nodes of the previously grown random forests is unnecessary, and
- as the principal idea is to separate faces of different identities, the classification results and the identity nomination will be the same.

It is to be noted that the "detection" random forests have already separated face images of different identities along the tree whilst finding discriminative Haar features between the face and non-face training data. By growing another random forest, we do not cannibalize those divisions. However, given the simplicity of growing a new forest and low training time [1, 21, 22, 9, 6, 10], we limit our discussions to the latter approach.

Identification and verification is straightforward once the identity threshold is calculated. If the threshold exceeds a heuristically chosen limit, the face is *identified* as one of P identities or as an unknown identity. If the identified identity matches the alleged identity, the face is *verified*.

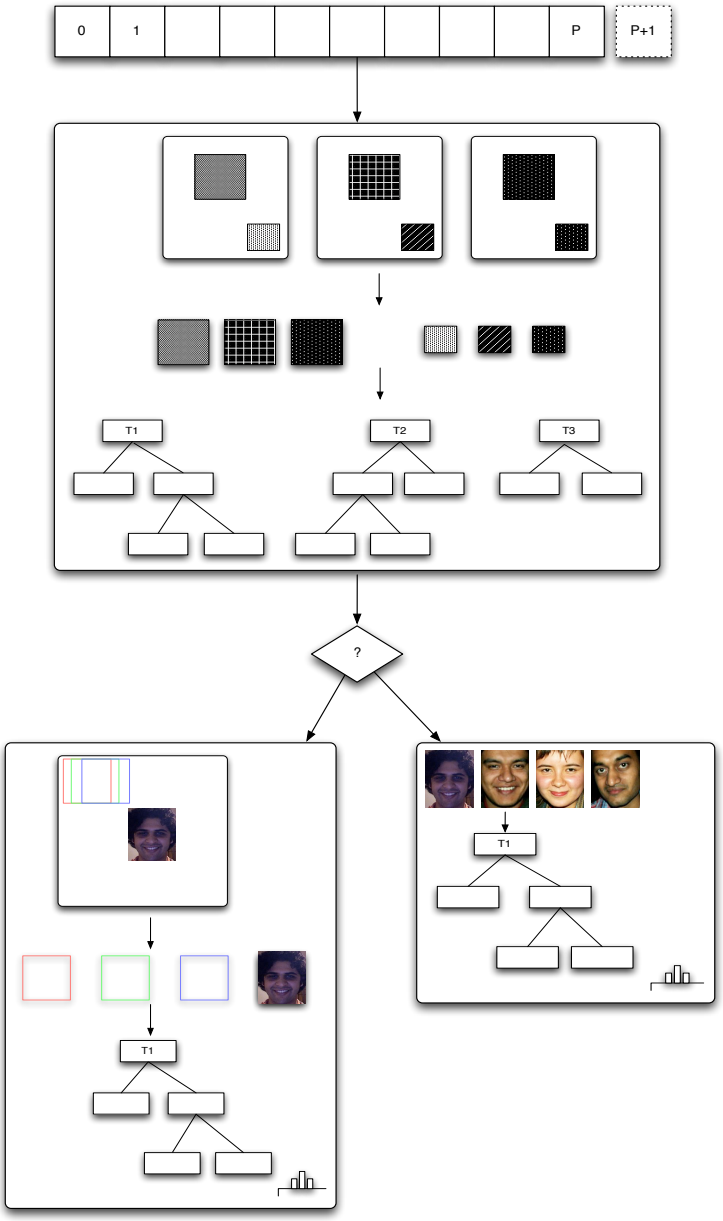

Figure 5.8: The complete framework of using random forests for face detection and recognition. The P class training data, 2 for detection and $P+1$ for recognition, is used to train random trees. A sliding window mechanism returns the detected faces in the input image. A classification stage returns the identity of the face (see text).

We conclude the recognition framework by considering the four scenarios outlined by Turk and Pentland [39] (discussed in Section 4.2). The scenarios are modified to model our framework.

1. **Regions of the input image do not contain faces and do not share any similarity with faces**. Our detection framework discards all subwindows that do not contain faces. Such subwindows are not considered further.
2. **Regions of the input image do not contain faces but share a few local visual descriptors with faces**. The random forest iteratively selects discriminatory features to divide faces and non-faces. If a non-face rectangular region shares a few geometric features with faces, along the depth of tree features are found that subsequently separate faces and non-faces.
3. **Regions of the input image contain faces but are not similar to any of the identities**. In the random forest grown for recognition, face images will be nominated as belonging to the $p + 1$ identity.
4. **Regions of the input image contain faces and is similar to identity** p. The identity likelihood will nominate the face as belonging to identity p.

Figure 5.8 describes the overall system. A random forests is grown for detection using the two class training data. A random forest is grown for recognition using the $P + 1$ class training data. The sliding window based detection algorithm returns the location and extent of detected faces in the input image. Each face is now propagated down the forest and an identity is returned based on confidence analyses.

5.8 Face Learning

As part of our motivation, we discussed that if a human face was not recognized as one of the known identities then the robot introduces itself and learns the new identity. This characteristic renders a human-like experience and improves the social interaction between humans and robots.

We model face learning also by growing the random trees further. Let us assume the mobile robot has introduced itself to a human being ($\Omega_{p'}$) and images are captured. As it is likely that the images contain faces in backgrounds, we initiate the learning stage with the detection and segmentation of the faces from the backgrounds. The segmented faces are the training data for $\Omega_{p'}$.

Each segmented face is propagated on a tree. At the leaf node, the trees are grown further till the new identity is separated from the remaining identities at the leaf node. The importance of this step is that previously, the random trees for recognition were grown until images from only a single identity remained in the leaf nodes or to a heuristically

chosen depth. By growing the trees further till the new image is separated respects the classification accuracy of the forest before the new identity was introduced.

Subsequent to this procedure on all segmented faces, the framework is considered to have *learnt* the identity. The recognition framework remains intact and identification or verification is carried out as before.

We had previously discussed a simplification to growing the random forests for detection further to also recognize by building a separate forest only for recognition. A similar simplification can be extended here. If the number of identities is not large, the random forests built for the $P + 1$ identities can be rebuilt for $P + 2$ identities[2]. Thus, we do not need to grow the forest trained on recognition further but simply rebuild the same with the newly segmented faces as an additional identity.

We conclude our framework with the observation that it is possible to build only a single random forest for face detection, face recognition and face learning, analogous to Deselaers et al. [14]. Our separation of concerns approach, i.e. independently developing the three tasks, allows for better parameterization and implementation simplicity.

5.9 Parameters

As we deal with several heuristics and decision criteria, we present a review of a few important parameters in this section. We enumerate the same followed by the values found optimal in our experimental setups.

1. **Forest Size.** This specifies the number of random trees to grow. As we discussed in Section 5.1, increasing the number of trees improves the generalization accuracy. However, the improved accuracy is at the cost of an increase in the training and classification time. For detection, we build either 2 or 3 random trees. For recognition, we build between 3 and 20 trees. We note that results on recognition is faster as we only need to integrate classification results. On the other hand, with detection, a multi-resolution analysis needs to be performed on each of the random trees.
2. **Feature Size.** This is the maximum allowed size of sampled subwindows of the training data. We train on 24×24 images and found a parameter tune at 11 to be optimal. In previous experimental runs on images of other dimensions, such as 50×50, a value of 20 was found optimal. We estimate that values a little less than

[2]It is assumed that the unknown identity class does not contain any face images of identities that might be learnt by the mobile robot.

$\sqrt{M} \cdot 0.5$ would prove optimal. It was additionally found the detection accuracy lowered when larger limits were allowed.

3. **Minimum Feature Size.** This is the minimum allowed size of sampled subwindows. Preventing instantiation of tests that sampled windows of 1×1 pixels and 2×2 pixels improved detection results. We found a value of 4 to be optimal. In contrast, recognition results improved if this parameter was disabled.
4. **Haar Filter Set.** The feature extraction is an extensible architecture that can consider additional Haar filters. We used only four features but the framework allows an indefinite number. Pham and Cham [29] use 19 Haar filters.
5. **Total Tests.** We previously discussed that we quicken the tree growth by instantiating a number of tests and selecting the test that returns the best split in accordance to the Shannon Entropy calculation. We set the parameter to a value of 200.
6. **Node Count.** Limiting the number of nodes a random tree is grown to has many advantages. The first of which is that the memory requirements are reduced. The time taken for the training, sliding window and classification is reduced. As a collection of random trees is grown, the above advantages are important. Further, the FMP is a suitable probabilistic estimation to determine if a subwindow is to be nominated as a face. We found limiting the growth to 8000 nodes sufficient for a $(\{\mathbf{\Gamma}_f\}_1^{10000}, \{\mathbf{\Gamma}_{nf}\}_1^{10000})$ training data collection[3].
7. **Scale Set.** We previously discussed that to detect faces of all sizes, we use multiresolution analysis. We scale the image to different sizes and sample M dimensional subwindows. At some scale, a face in the input image is rendered M dimensional and detection is accomplished. Needless to say, the more the number of scales used, the higher is the precision of the alignment and bounding box. We consider a total of 13 scale factors, where

$$s = \{0.15, 0.25, 0.35, 0.5, 0.7, 1.0, 1.3, 2.0, 3.0, 3.25, 4.0, 4.7, 5.4\}.$$

8. **Sampling Offset.** We mentioned in Section 5.5 that to reduce an exhaustive sampling of subwindows that offset subwindows by 1 pixel, we can increase the offset. We keep the value at 2.
9. **Detection Window Merge Ratio.** As part of the module to merge windows, this parameter sets the minimal area of overlap between two detection windows before they are merged. The neighborhood range decides the detection windows to consider for merging. We discussed either using overlapping rectangles or calculating the center of a circle and finding other windows in range. The approach used to find neighborhood windows is relevant to the value this parameter takes.

[3] We previously discussed that, typically, the number of non-faces exceeds the number of faces in the training data as a larger group of entities constitute as backgrounds. As we train on large collections, the number of faces and background images are kept the same.

Depending on the value of $2R$, the diameter of the circular region, the area over which overlapping rectangles are found is larger. In this instance, the parameter must be set to a higher value in comparison to merging neighbors only on overlapping rectangles. Not doing so will merge many unrelated detections. We ran experimental set-ups with both approaches and found the best results when overlapping rectangles was used with the parameter set to 0.3.

10. **Face Map Threshold.** The FMP represents the likelihood that a subwindow classified in the leaf node is a face. We ran experimental set-ups with good performances on all values from 0.3 to 0.7 (in steps of 0.1) and a value of 0.5 as the best.

5.10 Implementation

In line with the theoretical foundations and framework of our thesis, we have implemented a face detection and face recognition system with sufficient flexibility to be extended for future work.

5.10.1 Overview

The implementation uses image processing operators and image data handling structures from the OpenCV library. OpenCV is a computer vision library developed by Intel. It is an open source software and freely available for download. The OpenCV development stresses on real-time image processing [84]. The overall framework of our implementation is described in Figure 5.9. The figure depicts the *frame grabber* component and a flow chart execution of the program.

The mobile robots we intend to the run the software on are equipped with Bumblebee®2 stereo vision cameras. The cameras capture an image and the image is stored in the memory with pixel values in the YUV space. The image is converted from the YUV space to the RGB space using the conversion matrix [45]

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.13983 \\ 1 & -0.39465 & -0.58060 \\ 1 & 2.03211 & 0 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix},$$

where the values of R, G and B are taken to be in the range $[0, 1]$. The data directly instantiates the main image manipulation data structure of OpenCV, `IplImage`. Specially, the read image is copied to the `imageData` member. The `IplImage` object returns the information (height, width, channels) necessary to execute further operations, such as the sliding window detection mechanism and cropping detected faces.

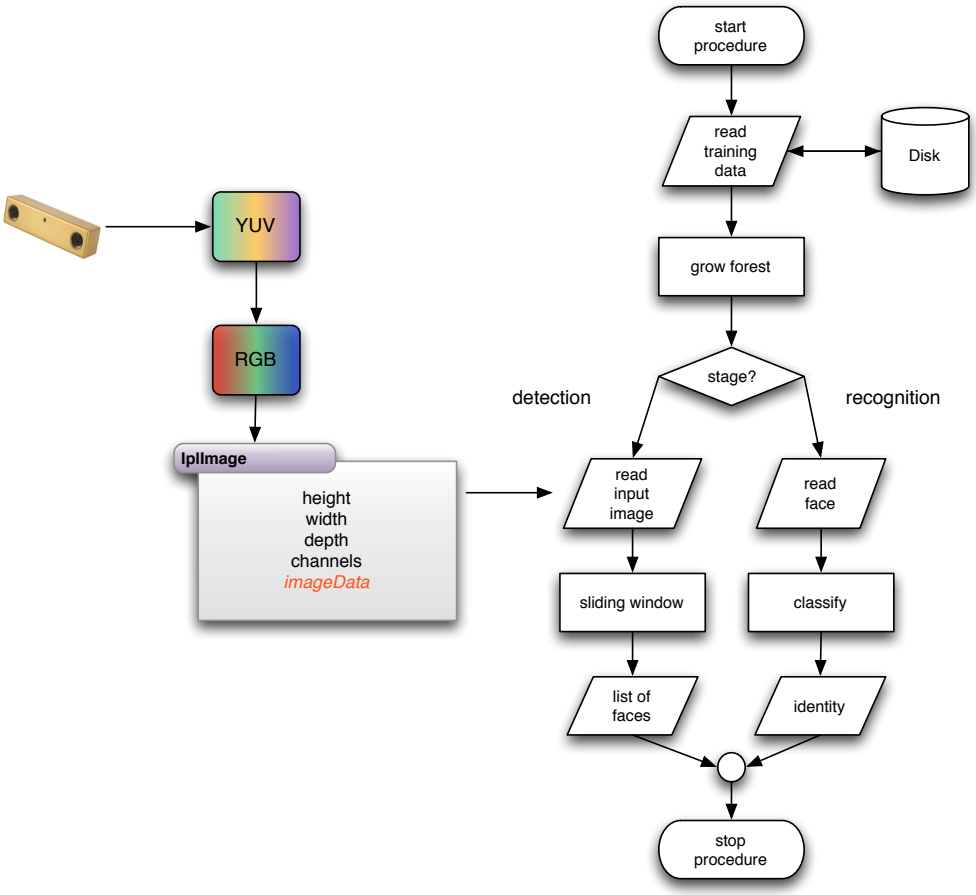

Figure 5.9: The figure depicts a flow chart of the complete system. There are two execution paths, one of detection and the other of classification. In the left-hand side of the figure, image capture and forwarding the image data is illustrated. OpenCV image data handling structures are used in the implementation (see text).

On the right side of the figure, a flow chart summarizes two execution paths of the system. The system reads the training images from the disk (the training images are assumed to be appropriately organized). As discussed previously, a face detection system expects two collections whilst a face recognition system expects $P + 1$ collections, where P is the number of identities and a separate set accounts for an unknown identity.

A forest is grown and the system can subsequently either run in the *detection mode* or the *classification mode*. If the execution proceeds in the detection mode, the input image is stored in the instance of `IplImage` described above and passed into the main framework. The sliding window detection mechanism module returns a list of bounding

boxes predicted to be faces. The faces are cropped and saved.

The framework is re-run for recognition. The detected face is propagated on the forest and the classification results are returned as the *identity* of the face. It is rarely the case that we constantly add images to any face detection training data. Similarly, the training data for recognition is augmented only in case a new identity is to be learnt. To prevent growing random forests at every run, we have implemented a module to save the random forests locally. The execution path of the framework does not require access to the training data but to the saved abstract alone.

To visualize grown trees, we use GRAPHVIZ [4], a graph visualization software. As we build binary trees, we create a digraph with the command:
`digraph g { node [shape = record,height=.1]`, where the parameters define the shape of the nodes and the default height. When a child node is instantiated in the random tree, we use the command:
`node%d[label = "<f0> %d| <f1> %s |<f2> %s "]`, where the first two parameters are unique numbers assigned to the node, the third parameter is the composition in terms of the training data with respect to this node and the fourth parameter is the entropy value of the best test chosen at the node. We assign a global count of the nodes created to the first two parameters. Figure 5.5 illustrates our design. In the figure, every node in the displayed random tree is characterized with a global count of the node, the composition and the entropy value of the best test. Subsequently, we monitor the skeleton of the growth, as shown in Figure 5.6, by setting the `style` attribute to `invis` and the `shape` attribute to `rectangle`. We note that the description of the visualization as specified above is drawn from the semantics of the DOT language [5].

5.10.2 Software Architecture

We present a general skeleton of our software architecture below. We segment the architecture into three parts: image manipulation structures, random tree structures and parameterization. The images, once read, are converted to the integral image representation discussed in Section 5.3.3. To access pixels easily, we wrote a wrapper for the `IplImage` data structure to use both tri-band images and grayscale images as shown in Figure 5.10. In the figure, the pixel values of the tri-band image or the grayscale images can be read with the `get` methods. The images are converted to an array of integers using the recursive definition detailed in Equation 5.10 and Equation 5.11. Each of the N training images are read and stored in the integral image representation as a vector called `VectorOfIntegralImages`.

[4] http://www.graphviz.org/
[5] http://www.graphviz.org/doc/info/lang.html

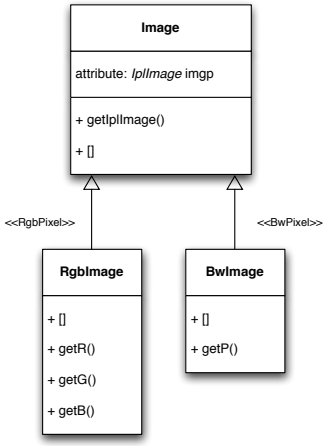

Figure 5.10: An implemented wrapper for the IplImage image manipulation data structure.

As we described in Algorithm 3, the vector at root node Γ_t is recursively split into the left and right children based on a test characterized by $T_t = \{x_t, y_t, h_t, w_t, \mathbf{\Gamma}_l, \mathbf{\Gamma}_r, \theta_t\}$. In Figure 5.11, the class `Test` returns the entropy H of a split using Equation 5.12.

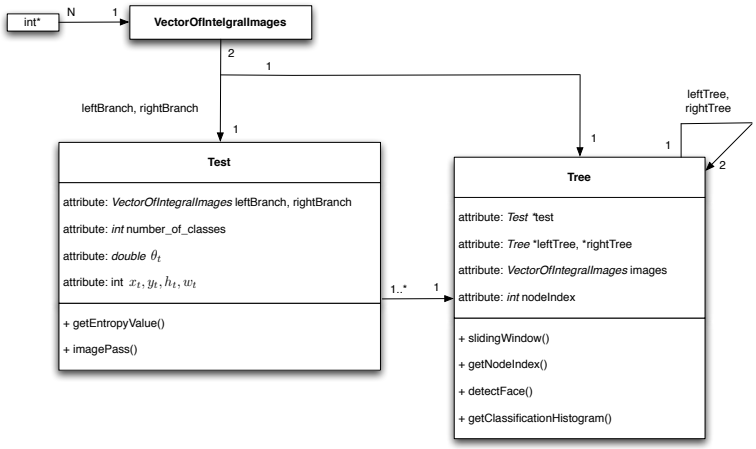

Figure 5.11: An overview of class relationships.

The module `imagePass` returns true if an image subwindow crosses the threshold and is categorized to the left node. Typically, there is only one test per node but we discussed the use of L nodes to shorten tree growth. This is expressed in the compositional relationship between class `Test` and class `Tree`.

To control the depth of the tree as the number of nodes grown, the `nodeIndex` attribute is used. In the detection execution path, the `detectFace` module samples the image at

different scales and the `slidingWindow` module iteratively detects faces. In the recognition execution path, the `getClassificationHistogram` returns the identity.

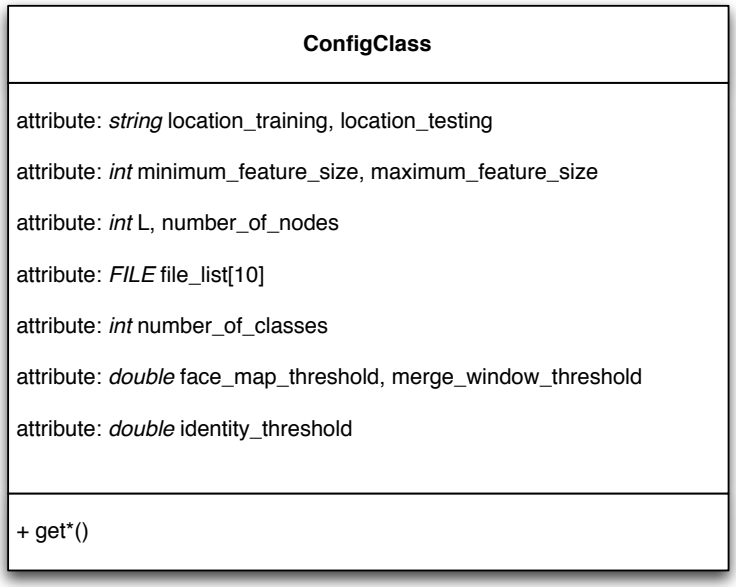

Figure 5.12: The `ConfigClass` data structure for the parameterization.

We have, to this point, described the image manipulation and the random tree software data structures. For our parameterization, we consider a separate structure titled `ConfigClass`. Figure 5.12 shows a subset of important attributes. The class is complemented with corresponding `get` modules. The attribute `file_list` contains the log files of the random forests. The main module operates on the basis of the overview presented in Figure 5.9. A collection of random trees are grown and the post processing step effectively aggregates the results from each tree.

6 EVALUATIONS

In this chapter, we present evaluations of our framework on face detection and recognition. We compare face detection with random forests to the adaptive boosting based face detector by Viola and Jones [30]. Our evaluations are presented with ROC curves in Section 6.1. We compare face recognition with random forests to support vector machines and log-linear models in Section 6.2. The latter evaluations are presented with confusion matrices and rank-1 recognition rates, both of which are introduced shortly.

6.1 Face Detection

In this section, we present evaluations on face detection with random forests. In Section 6.1.1, we introduce ROC curves. In Section 6.1.2 we present our comparisons.

6.1.1 Introduction

The evaluation of a face detection system [30] is typically presented using a receiver operating characteristics (ROC) curve [27]. The ROC curves is a plot of *sensitivity* versus *1 - specificity*. For a detection system, the number of faces correctly detected are identified as the *benefits* of the system. On the other hand, the number of false positives returned are identified as *costs*. ROC curves are influential in analyzing tradeoffs between benefits and costs.

The ROC curves are plotted with true positive rate (TPR) on the Y-axis and false positive rate (FPR) on the X-axis. To recapitulate, the true positive rate is calculated as,

$$\begin{aligned} \text{True Positive Rate} &= \frac{\text{true positives}}{\text{true positives + false negatives}} \\ &= \frac{\text{number of faces detected}}{\text{total number of faces in the input image}} \end{aligned} \tag{6.1}$$

where Equation 6.1 is the detection accuracy previously discussed in Chapter 3. This is the sensitivity of the detection system.

The false positive rate (FPR) is calculated as,

$$\text{FPR} = \frac{\text{false positives}}{\text{false positives + true negatives}}$$

$$= \frac{\text{number of subwindows incorrectly labeled as faces}}{\text{all sampled subwindows in the input image}} \tag{6.2}$$

where the cardinality of the sampled subwindows is usually drawn from the sliding window detection algorithm outlined in Algorithm 4.

The specificity is defined as $1 - FPR$. Thus, the ROC curve is equivalently sensitivity vs. 1 - specificity and TPR vs. FPR. A classifier, if allowed to make a random guess on the class labels then the performance in the ROC space is a diagonal line through the origin as shown in Figure 6.1.

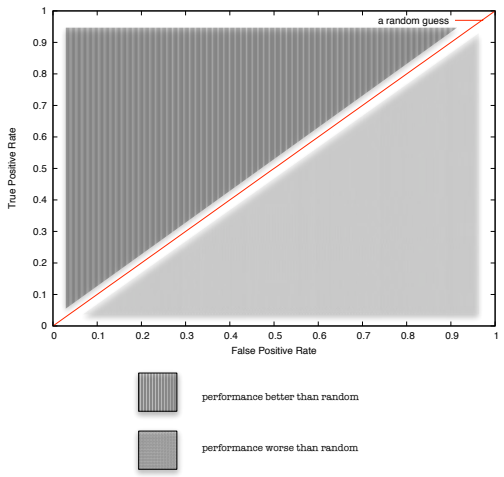

Figure 6.1: A ROC space. The diagonal represents a random decision by a binary classifier. The space above the diagonal represents a performance better than a random decision. The space below the diagonal represents a performance worse than random.

In accordance, a plot of the ROC curve of a classifier should be above the diagonal, as we intend to achieve performance better than random. The FPR calculated as above is approach specific as the total number of sampled windows is approach specific. For instance, if an approach uses a multi-resolution sampling like our framework then the number of scales considered and the offsets both contribute to the FPR.

Hence, face detection results are either presented with respect to the number of false positives or the relative rate of the false positives with respect to the number of faces

in the input image. In the latter, Equation 6.2 is rewritten as,

$$\text{FPR} = \frac{\text{number of subwindows incorrectly labeled as faces}}{\text{total number of faces in the input image}} \tag{6.3}$$

It can be derived that the FPR calculated thus is in the range $[0 : Z]$, where

$$Z = \frac{\text{all sampled subwindows}}{\text{total number of faces in the input image}}$$

and Z is the right most point in the X-axis of the ROC curve plot.

ROC curves help us measure the highest achievable detection accuracy with respect to an allowed number of false positives. Global parameters, such as the number of nodes grown and the face map threshold, are varied to measure the detection accuracy and the accompanying false positives count. The responses are plotted. If the algorithm allows every subwindow to be classified as a face then the FPR will take the maximum value but the TPR will be 1.0. This is the final point in the ROC curve where we have 100% detection accuracy at the cost of every non-face falsely labeled as a face.

6.1.2 ROC Curve Plots

We evaluate the performance of our detection system against the adaptive boosting based face detector proposed by Viola and Jones [30]. The Viola and Jones detector offers one of the most impressive detection performances [29, 30]. It is also freely available as a module of the OpenCV library. Configuration files trained to detect faces are also supplied as part of the OpenCV installation. We note that the supplied configuration files offer the best results.

A clear comparison of the two approaches is only possible under the following conditions:

- the approaches are trained on the same training data and
- the results are evaluated under similar conditions on the same test data.

In accordance, we ran experimental analyses on a comprehensive set of training data freely available (enumerated in Section 4.3.1). The best performances on both random forests and the adaboost approach was achieved on the Universidad Politecnica de Valencia Face database [65]. We consider three subsets for training:

1. $\mathbf{C}_1$. The subset has a random selection of 1000 faces and 1000 background images.
2. $\mathbf{C}_2$. The subset has a random selection of 4000 faces and 4000 background images.
3. $\mathbf{C}_3$. The subset has a random selection of 10000 faces and 10000 background images.

The non-faces in each of the subsets were obtained by extracting random patches from the images that do not contain any faces and are typical backgrounds. All the training images used are 24×24 pixels in height and width.

The test set is composed of a difficult subset and a randomly chosen subset drawn from both the CMU test collection (Section 4.3.2) and the BioID face data set (Figure 4.8). The former is composed of faces of varying sizes in complex backgrounds and different illumination conditions as well as rotated faces. The latter was chosen as it approximately presents typical images that a mobile robot may encounter. The images in the BioID dataset contain faces with the following characteristics:

- slight variations in pose,
- sufficient variations in expressions,
- slight variations in illumination,
- slight variations in the dimension of the faces across the identities and
- typical backgrounds that a mobile robot might encounter such as windows and the backdrop of a room.

The complete test collection consists of 123 images with a total of 188 faces. With respect to this test set, we present the following evaluations for face detection:

1. $\mathbf{R}_1$, a ROC curve plot of random forests vs. the adaboost module, both trained on C_1.
2. $\mathbf{R}_2$, a ROC curve plot of random forests vs. the adaboost module, both trained on C_2.
3. $\mathbf{R}_3$, a ROC curve plot of random forests vs. the adaboost module, both trained on C_3 and the adaboost trained for 6 stages. We refer to the corresponding adaboost training as *Adaptive Boost #1*.
4. $\mathbf{R}_4$, a ROC curve plot of random forests vs. the adaboost module, both trained on C_3 and the adaboost trained for 9 stages. We refer to the corresponding adaboost training as *Adaptive Boost #2*.
5. $\mathbf{R}_5$, a ROC curve plot of random forests trained on C_3 vs. the adaboost module used with a configuration file supplied as is. We refer to the corresponding adaboost training as *Adaptive Boost #3*.

An important disparity between the approaches is the training time. As we discussed earlier, the adaboost face detector builds cascades of binary classifiers. Each attentional cascade reduces the false positive rate further. In accordance, the training time increases for each additional stage. Training the adaboost module for R_3 to 6 stages requires 5 days on an Intel Core 2 Duo processor equipped with 2 Gigabytes of RAM. Training the adaboost module for R_3 to 8 stages requires 7 days on an Opteron machine equipped with 16 Gigabytes of RAM. In comparison, training random forests on R_3 for a node

count of 10000 with $L = 200$ requires 400 seconds on an Intel Core 2 Duo processor with 2 Gigabytes of RAM[1].

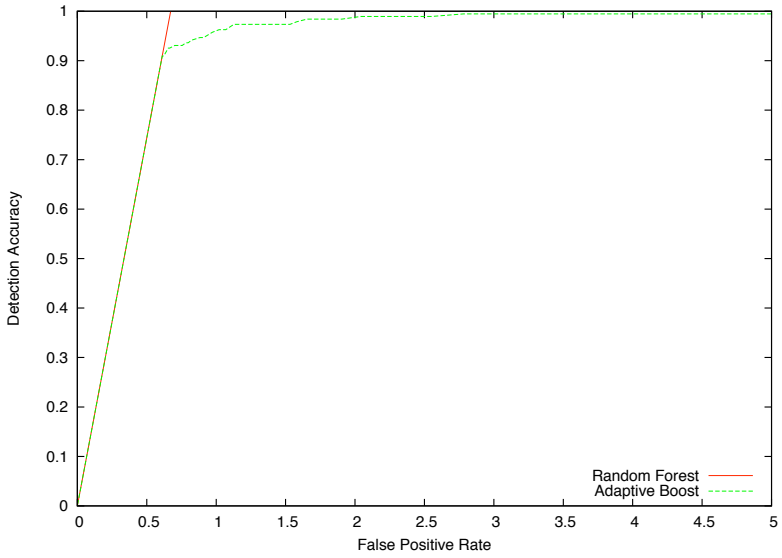

Figure 6.2: R_1, a ROC curve plot of random forests and adaboost, both trained on C_1. Random forests perform comparatively similar to adaboost.

R_1 is presented in Figure 6.2 with the adaboost module trained on C_1 for 4 days. It is clear from the figure that the random forests perform with similar detection accuracy with respect to false positive rates. However, for detection accuracies a little over 90%, the growth of false positives is faster with adaboost.

R_2 is presented in Figure 6.3 with the adaboost module trained on C_2 for 6 days. It is clear from the figure that for a little over 90% face detection accuracy, the false positive rate is considerably lower with random forests. Further, the detection accuracy is also lower than that of random forests prior to $FPR = 1.0$. Comparing the adaboost performance in Figure 6.2 and Figure 6.3, it can be observed that the detection accuracy is higher when trained on C_2. The larger set of faces and non-faces detect more faces.

R_3 and R_4 are presented in Figure 6.4 and Figure 6.5 respectively. It is clear that many stages need to be trained for the adaboost. We see a rapid drop in the false positive rate with an increase of just two stages. We recapitulate that the former was run for 6 stages

[1]The training durations have broader implications. As both random forests and adaboost approaches are generic, they are used equivalently for detection and classification of any object. Random forests offer a comparatively faster training phase.

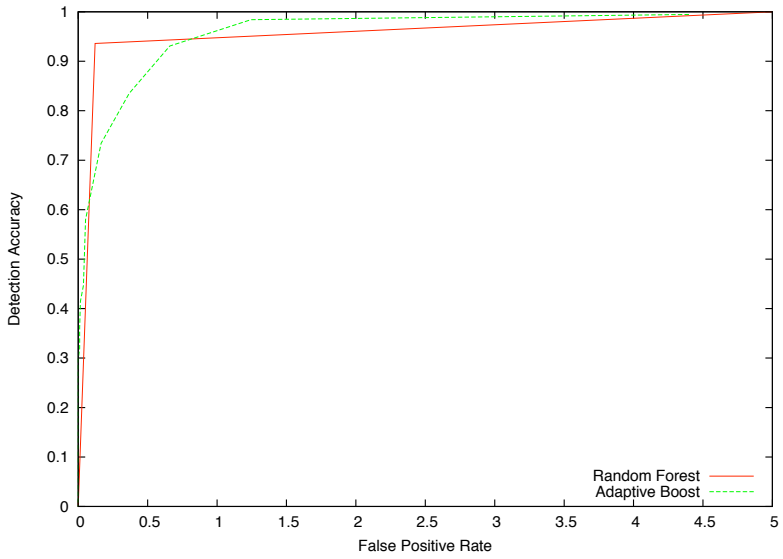

Figure 6.3: R_2, a ROC curve plot of random forests and adaboost, both trained on C_2. Random forests present a higher detection accuracy for lower false positives.

and the latter for 8. Random forests slightly outperform the adaboost approaches till approximately 91% detection accuracy. It is reported that the false positive rate is expected to drop to very low values with as many as 25 stages [2].

We finally present R_5 in Figure 6.6. From the figure, we observe a slightly better performance of Adaptive Boost #3 with respect to random forests. The random forests, on the other hand, perform better than Adaptive Boost #2 and Adaptive Boost #1, as was concluded from Figure 6.2 and Figure 6.3. We reiterate that the adaboost approach offers the best face detection system [30].

Figure 6.7 highlights the detection rates at a higher granularity for a FPR range of [0:1]. In the figure, the plot of Adaptive Boost #3 is close to the Y-axis in the figure. This is attributed to the fact that for low detection accuracies, Adaptive Boost #3 yields very low false positives. It is to be noted that neither Adaptive Boost #1 nor Adaptive Boost #2 perform similarly for the stages we trained them to. It can be concluded from the graph that for a detection accuracy of over 90%, the false positive rates are roughly equal for both Adaptive Boost #1 and random forests. For instance, allowing 25 false

[2] http://groups.google.com/group/OpenCV

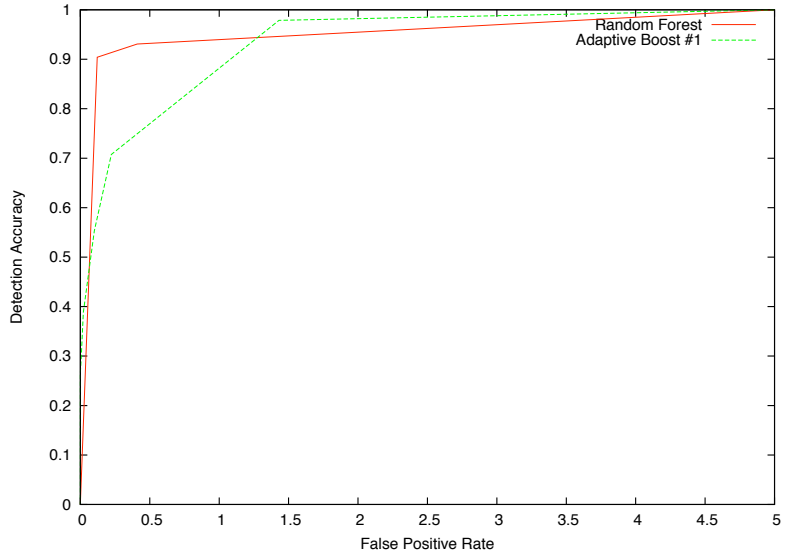

Figure 6.4: R_3, a ROC curve plot of random forests and adaboost, both trained on C_3. The adaboost was trained to 6 stages.

positives, Adaptive Boost #3 detects 178 and random forests detect 175 out of the 188 faces.

One of the principal justifications for the slightly lower performance of random forest when compared to Adaptive Boost #3 is that of the post processing phase. The post processing phase must effectively merge the detections from both the multi-resolution sampling and all the random trees. We grow only three random trees for face detection as we observed the faces get successfully detected in each of the trees. Taking all of the merging strategies discussed in Section 5.6 into account, we found the following steps to be most effective:

- The individual scale results are merged by considering neighbors from overlapping rectangles as outlined in Figure 5.7. The merging procedure used is one that considers a weighted combination described in Equation 5.15. The α tied to the detection window with the highest FMP is set to 1 and the remaining are set to very low values.
- The results of the step above are scaled back to be applicable to the input image. Neighbors are found from overlapping rectangles as above. The biggest rectangle is selected and the others are discarded.

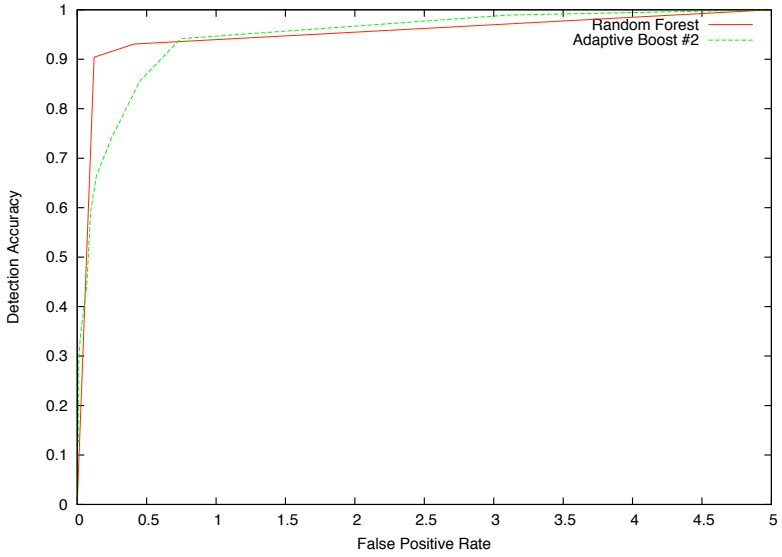

Figure 6.5: R_4, a ROC curve plot of random forests and adaboost, both trained on C_3. The adaboost was trained to 8 stages.

The above merging strategy, although it performs best among other strategies experimented with, is error prone. In comparison, the adaboost approach builds a cascade of classifiers to reduce the false positives in the attentional stages. The initial stages, on the other hand, are very liberal to allow for a large number of detections. There seems to be no trivial procedure to accomplish the same for random forests in short of a direct analogy: growing stages of random forests and forwarding detections from the previous stage to the next. Additionally, the detection subwindows suffer from alignment errors.

Figure 6.8 presents a few of our detections. Minor alignment errors are present in some of the detections. However, if we consider the image highlighted with a blue border, we note that the second detection (orange subwindow) is badly aligned[3]. In the image highlighted with a turquoise border, we note a false positive. In the image highlighted with a purple border, we note a false negative, i.e. a face is not detected at all.

We aim to the address the above in our future work (Chapter 7) and conclude with the remark that random forests offer extremely competitive results with respect to the best face detection system in literature.

[3]Our evaluations do not nominate the same as a correctly detected face.

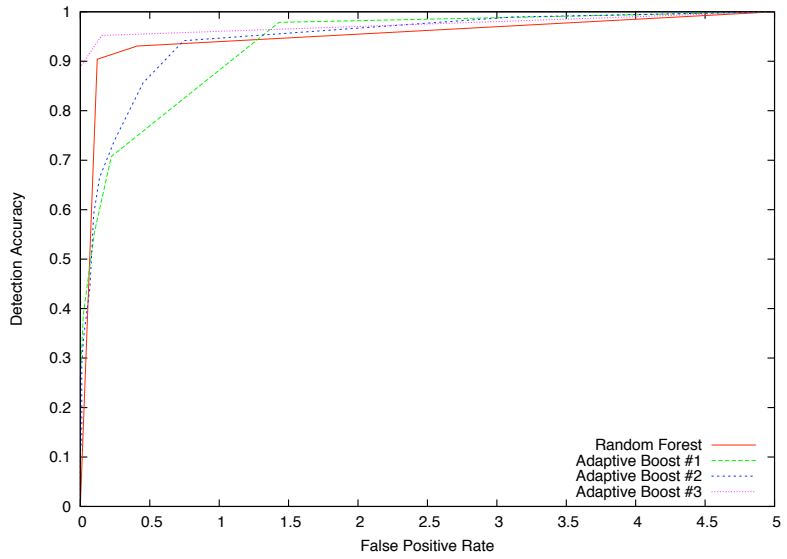

Figure 6.6: R_5, a ROC curve plot of random forests and adaboost modules. Adaptive Boost #1 is adaboost trained on C_3 to 6 stages. Adaptive Boost #2 is adaboost trained on C_4 to 8 stages. Adaptive Boost #3 is the performance with a configuration file for face detection provided as part of the OpenCV installation.

6.2 Face Recognition

To present our face recognition results we use the simplified construction of our recognition framework. Namely, we build a separate random forest trained on the P class collection. We present all our evaluations on 4000 nodes and $L = 1000$. We report that growing a random tree to 4000 nodes with $L = 1000$ requires 75 seconds. Growing a random tree to 1000 nodes with $L = 200$ requires 3 seconds.

6.2.1 Testing Framework

In Chapter 4, we discussed the FERET evaluation as one of the benchmarks used by most face recognition systems. However, the FERET evaluation is composed primarily of one training image and one probe image. Due to the training nature of random forests, we need more training images. Thus, we present three separate evaluation collections below. Needless to say, the training data D and the test data T are completely

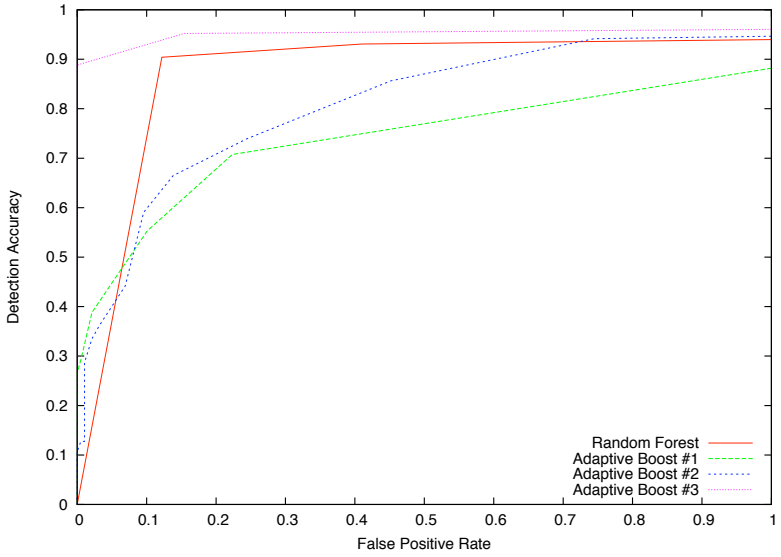

Figure 6.7: R_5, a ROC curve plot of random forests and adaboost modules. The plot is limited in the FPR range of [0:1].

disjoint, i.e. $D \cap T = \emptyset$.

- $\mathbf{T}_1$, 4 identities randomly drawn from the Universidad Politecnica de Valencia Face database [65] database with the following training data composition is $D = \{(188,0),(164,1),(92,2),(92,3)\}$, where D is the training data that has 188 faces for identity 0, 164 faces for identity 1 etc. The test data composition is $T = \{(14,0),(14,1),(14,2),(14,3)\}$.
- $\mathbf{T}_2$, 7 identities randomly drawn from the BioID database. We are not aware of any identity annotation of the BioID face database. We thus manually separated the database based on different identities and selected seven identities with the highest number of images. The training data composition is $D = \{(65,0),(59,1),(67,2),(72,3),(37,4),(65,5),(99,6)\}$. The test data composition is $T = \{(8,0),(2,1),(2,2),(5,3),(15,4),(9,5),(0,6)\}$. We note that a subset of the faces in T_2 were supplied with their respective backgrounds as part of the test collection for the ROC plots presented previously.
- $\mathbf{T}_3$, to probe the performance of our recognition framework when the number of training are less, we consider five randomly chosen identities from the Yale Database [67]. The training data composition is $D = \{(9,0),(7,1),(8,2),(9,3),(9,4)\}$. The test data composition is $T = \{(2,0),(4,1),(3,2),(2,3),(2,4)\}$.

Figure 6.8: Sample detections with random forest. We note the presence of a few minor alignment errors. Additionally, the image highlighted with a blue border reports a very bad alignment. The image with the turquoise border includes a false positive. The image with the purple border includes a false negative.

6.2.2 Rank-1 Recognition Rates

We present our results in terms of the rank-1 recognition rates and a *confusion matrix*. The rank-N recognition is the percentage that the correct label of the input image in within the top N suggestions [58]. Rank-1 recognition is, thus, essentially the results returned as is by the random forests using majority voting.

The confusion matrix is a table that contains information on the actual and suggested class labels [85]. For the three evaluations presented above, we draw the confusion matrices below. A confusion matrix is useful in capturing cases when an identity is repeatedly classified as another.

Table 6.1 presents the confusion matrix on the evaluation T_1. We achieve a rank-1

	0	**1**	**2**	**3**
0	14	0	0	0
1	0	14	0	0
2	6	0	8	0
3	1	0	0	13

Table 6.1: A confusion matrix on the evaluation collection T_1. Rank-1 recognition rate of 87.5%.

	0	**1**	**2**	**3**	**4**	**5**
0	5	0	0	0	0	3
1	1	1	0	0	0	0
2	0	0	2	0	0	0
3	0	0	0	5	0	0
4	1	0	0	0	14	0
5	0	0	0	0	0	5

Table 6.2: A confusion matrix on the evaluation collection T_2. Rank-1 recognition rate of 86.5%.

recognition rate of 87.5%. The rank-1 recognition can be calculated directly from the confusion matrix using:

$$\text{rank-1 recognition} = \frac{\text{sum of the diagonal elements}}{\text{sum of the rows or sum of the columns}}$$

Table 6.2 presents the confusion matrix on the evaluation T_2. We achieve a rank-1 recognition rate of 86.5%. Finally, the evaluation of random forests with very little training data is presented in Table 6.3. We achieve a rank-1 recognition rate of 85% where 11 of the 13 test images are correctly classified.

	0	**1**	**2**	**3**	**4**
0	2	0	0	0	0
1	1	3	0	0	0
2	0	1	2	0	0
3	0	0	0	2	0
4	0	0	0	0	2

Table 6.3: A confusion matrix on the evaluation collection T_3. Rank-1 recognition rate of 85.0%.

	0	1	2	3
0	14	0	0	0
1	0	14	0	0
2	0	3	11	0
3	0	0	0	14

	0	1	2	3
0	14	0	0	0
1	0	14	0	0
2	0	2	12	0
3	0	0	0	14

Figure 6.9: *Left*:T_1 with log-linear models with rank-1 recognition of 94.6%. *Right*:T_1 with SVM with rank-1 recognition of 96.4%.

	0	1	2	3	4	5
0	8	0	0	0	0	0
1	1	1	0	0	0	0
2	0	0	1	0	1	0
3	0	0	0	5	0	0
4	0	0	0	0	15	0
5	0	0	0	0	0	5

	0	1	2	3	4	5
0	8	0	0	0	0	0
1	1	1	0	0	0	0
2	0	0	2	0	0	0
3	0	0	0	5	0	0
4	0	0	0	0	15	0
5	0	0	0	0	0	5

Figure 6.10: *Left*:T_2 with log-linear models with rank-1 recognition of 94.6%. *Right*:T_1 with SVM with rank-1 recognition of 97.3%.

To compare our evaluations to important approaches used in literature, we consider face recognition using support vector machines and log-linear models. Support vector machines were previously presented in Section 4.2 and the corresponding illustration in Figure 4.3. We use the LIBSVM [62, 63] library for support vector machine based recognition. The values for C and γ were drawn using cross-validation. The values obtained thus are used to build $P(P-1)/2$ SVM with the complete training data. We use an implementation by Deselaers et al. [86] for building our log linear models. We present the confusion matrices and corresponding recognition rates here.

Figure 6.9 presents the rank-1 recognition rates on T_1 using log-linear models and SVM. Figure 6.10 presents the rank-1 recognition rates on T_2 using log-linear models and

	0	1	2	3	4
0	2	0	0	0	0
1	0	4	0	0	0
2	0	0	3	0	0
3	0	0	0	2	0
4	0	0	0	0	2

	0	1	2	3	4
0	2	0	0	0	0
1	0	4	0	0	0
2	0	1	2	0	0
3	0	0	0	2	0
4	0	0	0	0	2

Figure 6.11: *Left*:T_3 with log-linear models with rank-1 recognition of 100.0%. *Right*:T_3 with SVM with rank-1 recognition of 92.3%.

	$\mathbf{T}_1$	$\mathbf{T}_2$	$\mathbf{T}_3$
RF	87.5	86.5	85.0
Log-linear	94.6	94.6	100.0
SVM	96.4	97.3	92.3

Table 6.4: Rank-1 recognition rates with random forests (RF), log-linear models and support vector machines (SVM) on T_1,T_2 and T_3. The rates are expressed as percentages.

SVM. Figure 6.11 presents the rank-1 recognition rates on T_3 using log-linear models and SVM. From the above, we can conclude that both log-linear models and SVM perform, on an average, 9% better than random forests. As support vector machines offer one of the best face recognition results [61], it is to be noted that our rank-1 recognition rates are very competitive. We present a histogram for comparison of the rank-1 recognition rates in Figure 6.12 and Table 6.4.

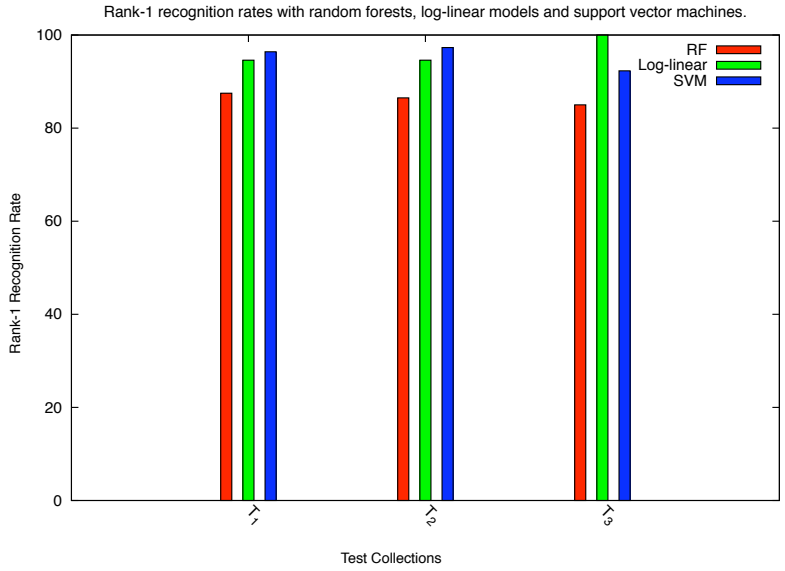

Figure 6.12: A histogram plot of the rank-1 recognition rates with random forests (RF), log-linear models and support vector machines (SVM) on T_1,T_2 and T_3. Random forest recognition rates are competitive.

Typically, recognition approaches present results on test images that are perfectly normalized. The face images are identical in the position of eye, nose and other features,

both in the training data and the test collection. However, in a mobile robot, we can not expect perfectly normalized faces. As a result, the test images in T_1, T_2 and T_3 were obtained automatically using Adaptive Boost #3. We discussed previously in Section 6.1.2 that although random forests detect faces comparatively to adaboost on the ROC curve plots, the former suffers with considerable alignment errors. As our recognition models expect the location of facial features at approximately relative spatial locations, the recognition rates are low when faces directly cropped from random forest detections are used. Jones and Viola [58] report that the faces detected using the adaptive boost face detector are additionally normalized with fixed eye positions. Our results above, however, are directly drawn from the extent and position as returned by Adaptive Boost #3.

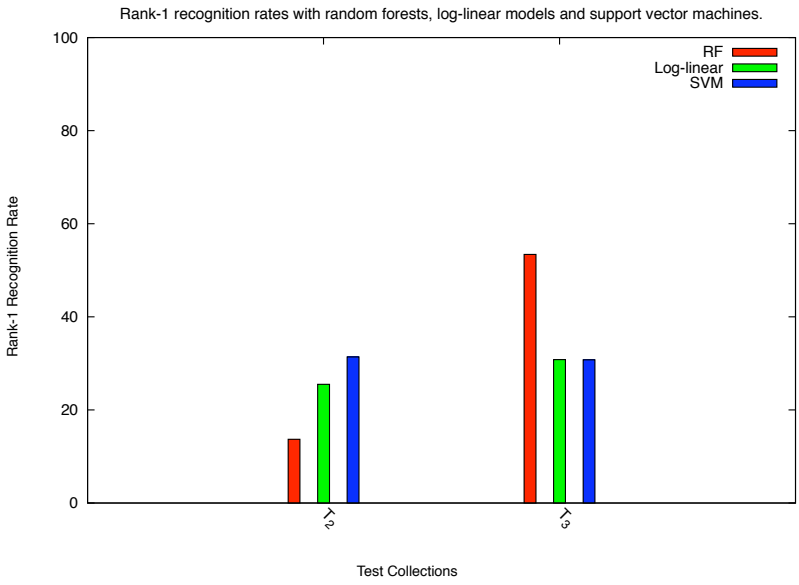

Figure 6.13: A histogram plot of the rank-1 recognition rates with random forests (RF), log-linear models and support vector machines (SVM) on T_2 and T_3. The test data was drawn from detections as returned by the random forest detection framework presented in Section 5.5.

The histogram in Figure 6.13 reflects the recognition performance on using the detections from random forests directly. We present rank-1 recognition rates on only T_2 and T_3 as all the images from T_1 are completely normalized. We reiterate that the training images are the same as those used for Figure 6.12. The results are also presented as a table in Table 6.5. Here, we observe that random forests perform well on T_3 as compared to T_2. This is attributed to the fact that the detection in T_2 suffer from higher alignment

	$\mathbf{T}_2$	$\mathbf{T}_3$
RF	13.7	53.4
Log-linear	25.5	30.8
SVM	31.37	30.76

Table 6.5: Rank-1 recognition rates with random forests (RF), log-linear models and support vector machines (SVM) on T_2 and T_3. The test data was drawn from detections as returned by the random forest detection framework presented in Section 5.5. The rates are expressed as percentages.

error rates.

6.2.3 Unknown Identity

Face recognition literature does not clearly model the identification of an unknown identity. The task is challenging as most approaches, typically, search for the likelihood of an input face recognized as an identity Ω_p. The result of this search is one of the known identities. A heuristically chosen threshold is introduced to nominate the input face as the corresponding identity or as unknown. Turk and Pentland [39], for instance, introduce a threshold with which the vector of weights from the eigenface framework are compared to determine if the face is known or unknown. We evaluate an approach to the identification of unknown identities with random forests below.

In Section 5.7, we outlined a procedure to *identify* strangers. We model our recognition framework as a $P + 1$ class problem where the additional class contains a collection of faces images of different identities. If an input face is recognized as belonging to this class, we designate the face to be unknown. We present our evaluations on T_1. Our test approach is as follows. To begin with, T_1 is composed of training images and test images of four identities. We created a separate collection of 188 face images of identities randomly sampled from the entire set of identities not considered in $\{\Omega_0, \Omega_1, \Omega_2, \Omega_3\}$. We denote the same as Ω_{P+1}. We now iteratively replace one of the four identities with Ω_{P+1} and measured the corresponding recognition rates. For instance, in the first run, we replace the training images for Ω_0 with Ω_{P+1}. We now measure the (i) rank-1 recognition on the test images for $\Omega_{1,2,3}$ and (ii) rank-1 recognition on the test image for Ω_0. The first measurement, $\mathbf{m}_1$, investigates the decrease in recognition on introducing a training collection with a wide variety of face images. The second measurement, $\mathbf{m}_2$, investigates the nomination of an unknown face as an unknown identity, in this case, Ω_0. Thus, in our experimental framework, an ideal recognition would categorize each test image in the exact same training data index.

Figure 6.14 plots the results. In the figure, the red color plot denotes that the training data composition is $\{\Omega_{P+1}, \Omega_1, \Omega_2, \Omega_3\}$ (there is no training data for identity 0). Test

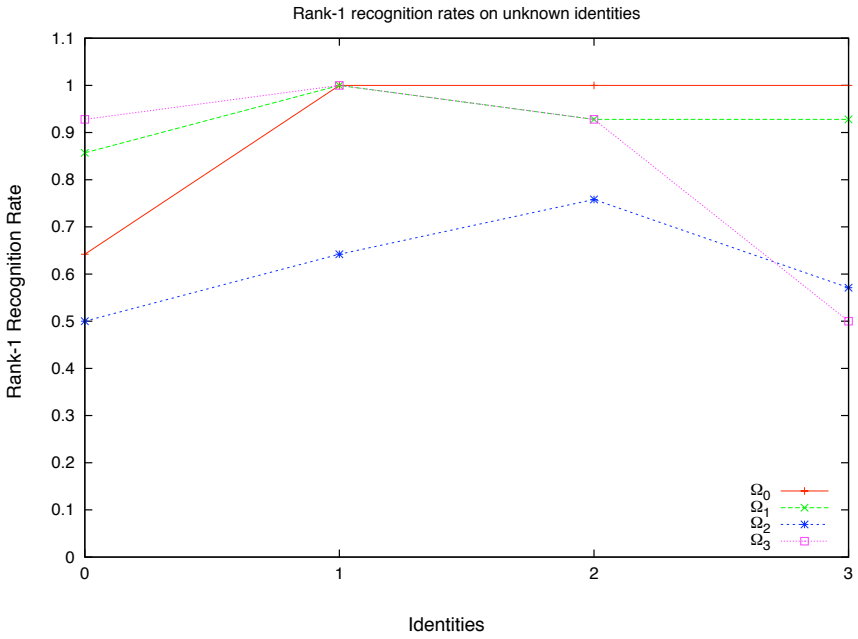

Figure 6.14: Rank-1 recognition rates unknown identities.

	Ω_0	Ω_1	Ω_2	Ω_3
0	64.2	85.7	50	92.8
1	100	100	64.2	100
2	100	92.8	75.8	92.8
3	100	92.8	57.1	50

Table 6.6: Rank-1 recognition rates with unknown identities. In row j, the corresponding identity is an unknown identity as the training data for Ω_j is replaced with Ω_{P+1} (see text).

images of Ω_0 was correctly recognized as an unknown identity (Ω_{P+1}) with a 64.2% accuracy. Correspondingly, test images for Ω_1 was correctly recognized as identity Ω_1 with a 85.7% accuracy. Test images for Ω_2 was correctly recognized as identity Ω_2 with a 50% accuracy and so on. Table 6.6 presents the corresponding rates.

The non-diagonal elements of the recognition in Table 6.6 denote $\mathbf{m}_1$. They indicate, on an average, the correct recognition of identities on the introduction of Ω_{P+1}, a collection with a wide variety of face images. From the table, we obtain 85.6%. The diagonal elements of the recognition in Table 6.6 denote $\mathbf{m}_2$. They indicate, on an average, an unknown face correctly identified as an unknown identity. From the table, we obtain 72.5%.

6.2.4 Forest Size

Marée et al. [5, 6, 9, 10] discuss the improvements in classification accuracy with an increase in the forest size. We proceed on similar lines but our evaluations are adjunct to face recognition. We probe rank-1 recognition with increases in the size of the random forest.

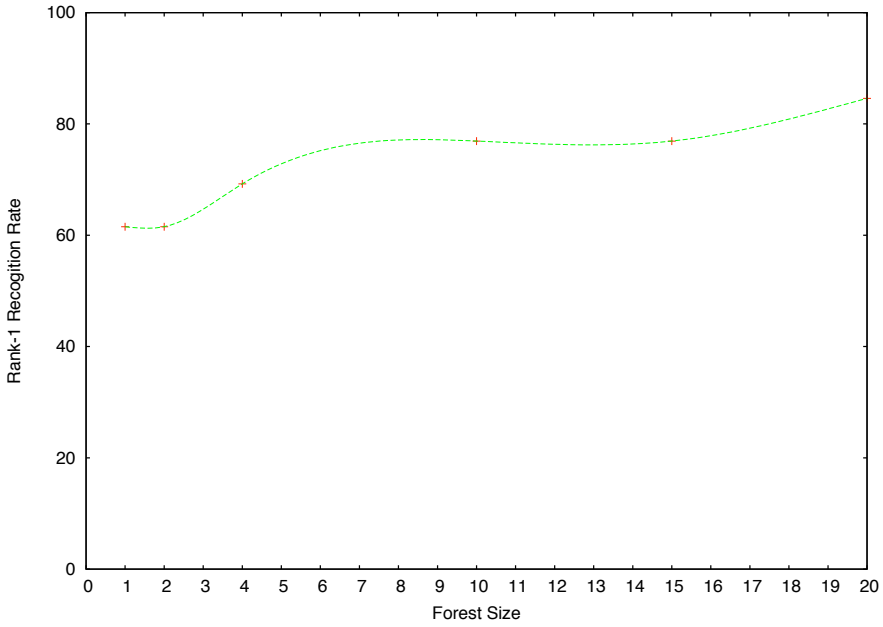

Figure 6.15: Rank-1 recognition rates with respect to the forest size. The recognition rates were measured with the random forest composed of 1, 2, 4, 10, 15 and 20 random trees.

Our initial experimental evaluation is on a single random tree. We consider $\{2, 4, 10, 15, 20\}$ random trees as voting predictors for the forest. Figure 6.15 presents rank-1 recognition rates on T_3. We observe that, with only 1 or 2 random trees, the performance is low with only a 61.53% rank-1 recognition rate. With 4 random trees, the random forest achieves a recognition rate of 69.23%. With 10 and 15 random trees, the rate stabilizes at 76.92%. Finally, at 20 random forests we achieve a rank-1 recognition of 84.61%. The corresponding confusion matrix for 20 random forests was presented earlier in Table 6.3.

	0	**1**	**2**	**3**
0	5	1	1	0
1	0	7	0	0
2	0	0	7	0
3	0	0	0	7

Table 6.7: A confusion matrix on KBSG Face Database. Rank-1 recognition rate of 92.86%.

6.2.5 KBSG Face Database

Our proposal and our motivation was towards a face processing system for mobile robots. The KBSG Face Database was presented in Figure 4.7. The database was created with the Firewire camera used with the RWTH Aachen Robocup@Home mobile robots [73]. The database is representative of the interaction between the robot and human subjects as the images are with different illumination conditions, typical backgrounds, side profiles, variable distances between the subjects and the camera.

We used Adaptive Boost #3 directly to crop the faces. We used the image processing libraries to universally crop and scale the faces to 48×48. The KBSG database consists of 4 identities with 30 faces each. We divide the collection to 23 faces for training and 7 faces for testing. We report that 26 of the 28 images were accurately recognized, i.e. a rank-1 recognition rate of 92.86%. The corresponding confusion matrix is presented in Table 6.7. We note that we do not present recognition on the KBSG Face Database with SVM and log-linear models as we are only interested in the accuracy achieved with our random forest implementation to be deployed on the RWTH RoboCup@Home robots. A comparison of recognition accuracies to SVM and log-linear models have already been reviewed earlier.

The results presented uses a forest of 20 random trees, with each tree grown to a depth of 3000 nodes. Further, we set $L = 1000$. The training time for the forest is 14 minutes. However, to learn new identities, we need a fast training framework, especially, to be applicable to the RoboCup@Home competitions. We thus grew only 3 random trees, each grown to a depth of 800 nodes and $L = 50$. We report the forest trains and returns recognition results in 1 millisecond. The resulting rank-1 recognition rate is 83.3%.

6.3 Summary

In this chapter, we presented comparative evaluations of face detection and recognition with random forests. Detection with random forests was compared to an adaptive

boosting based face detection systems. ROC curve plots were presented on detection with random forests and 3 adaptive boosting training configurations.

Recognition with random forests was compared to SVM and log-linear models. Rank-1 recognition rates were presented on three test collections. Additionally, rank-1 recognition rates with random forests were presented on the KBSG Face Database that captures typical image scenarios that a mobile robot will encounter.

Lastly, we presented improvements in classification accuracy on increasing the size of the forest. In the next chapter, we draw our conclusions. We briefly summarize the evaluations and discuss future work.

7 Conclusion and Future work

7.1 Conclusions

Our thesis presents an integrated framework to the detection, recognition and learning of faces with random forests. Random forests are structurally homogenous to classical decision trees but randomize the decision criteria. Random forests sample random rectangular regions and learn features from training data. We have an implementation of our framework that is compared to state-of-the-art approaches. Particularly, we compare the performance of face detection with random forests to adaptive boosting. Random forests outperform adaptive boosting when they are trained on the same data. Random forests perform comparatively to adaptive boosting using the training configuration supplied as part of the OpenCV installation.

We however note that face detection with random forests suffers from alignment errors. The alignment errors are not tied theoretically to random forests but lie in the error prone merging techniques developed. Merging is influenced by our search for the neighbor range. If the neighbor range is large, multiple faces in the immediate vicinity will be falsely considered to be the same face. If the neighbor range is small, multiple subwindows for the same face collected from the multi-resolution analysis and multiple trees will be falsely considered as multiple faces. Thus, we are typically left with a large number of nominated detections in neighboring locations making our post-processing non-trivial.

We presented comparisons of face recognition with random forests to face recognition with SVM and log-linear models. We considered 3 test collections and observed that random forests performed within a range of 9%, on average, to both SVM and log-linear models.

A critical characteristic of random forests is the training time. Due to the fact that completely random features are sampled, the random tree growth is fast. We reported in Section 6.2.5 that the training of 3 random trees to a depth of 800 nodes with $L = 50$ takes a millisecond. This feature allows the rebuilding of trees in feasible time. A critical parameter is the forest size. Allowing an increase in the number of trees grown increases the classification accuracy and training time. A tradeoff is necessary depending on the time constraints. We have shown that with 20 random trees, high classification rates are obtained.

We have also described two approaches to the realization of our overall framework. The first is building a random forest on the two-class face detection training data and growing the same further, for recognition and the introduction of a new identity. The second and simpler approach considers a separate forest for detection and a separate forest for recognition. On the introduction of a new identity, the forest for recognition is either rebuilt or grown further.

We present a brief overview of our results in Table 7.1. The table presents our concerns with the detection of faces with random forests and its comparative performance to adaptive boosting training configurations. Additionally, rank-1 recognition rates on the KBSG Face Database is 92.86%, rendering the same to be sufficiently accurate to be deployed.

<table>
<tr><th>Task</th><th>Tool</th><th>Conclusions</th></tr>
<tr><td>Face Detection</td><td>ROC Curve plot</td><td>From Section 6.1.2 and Figure 6.6:
RF outperform adaboost with same training data.
RF perform comparatively to adaboost with installed training configuration.
RF results present non trivial merging procedure.
RF suffer from alignment errors.
RF training is fast.</td></tr>
<tr><td>Face Recognition</td><td>Rank-1 recognition rates</td><td><table><tr><th></th><th>T_1</th><th>T_2</th><th>T_3</th></tr><tr><td>RF</td><td>87.5</td><td>86.5</td><td>85.0</td></tr><tr><td>Log-linear</td><td>94.6</td><td>94.6</td><td>100.0</td></tr><tr><td>SVM</td><td>96.4</td><td>97.3</td><td>92.3</td></tr></table></td></tr>
</table>

Table 7.1: A summary of the evaluations on face detection and recognition with random forests.

In summary, we have presented, reviewed and realized an integrated framework for the detection, recognition and learning of human faces with random forests for mobile robots which fulfills our requirements aimed at.

7.2 Future Work

Our work was motivated and is towards a face processing system on a mobile robot. While we present fast training times with random forests, we are yet to ascertain if a random forest framework offers the best run-time deployment. Specifically, we intend to verify the following:

- Does detection with random forests offer an acceptable detection time versus accuracy trade-off with the mobile robot operating in a dynamic environment? We intend to probe the possibility of achieving satisfactory and fast detection of human faces as the mobile robot is carrying out its typical tasks.
- Does recognition with random forests offer an acceptable recognition time versus accuracy trade-off in similar conditions as above?
- Is the face learning framework fast enough to render a natural experience? Specifically, we discussed our intention of the framework made applicable in the RoboCup@Home competitions. The RoboCup@Home necessitates a rapid learning of new identities. To recapitulate, the face learning framework can be realized either by growing the forest further or rebuilding the trees. If we realize the framework by rebuilding the trees, we intend to investigate a trade-off between the number of trees and the recognition accuracy in this scenario. If we realize the framework by growing the trees further, we intend to investigate a trade-off between the number of trees to grow further and the recognition accuracy.
- We intend to ascertain the benefits in terms of speed, simplicity, memory efficiency of a unified framework as opposed to individual implementations of face detection and recognition systems.

We previously discussed that the random forest framework is generic. It is thus an interesting extension to consider the detection and recognition of any general object, such as a cup, in complex backgrounds. The RWTH Aachen RoboCup@Home robots are already equipped with a speech recognition system [73, 87]. Relaying information from the speech component to recognize humans and objects is a very important extension.

We conclude with the fact that our integrated approach to the detection, recognition and learning of human faces can be modeled to the detection and recognition of any generic object. Further, the framework presented allows the possibility of including extensions that will lead to an improved social interaction between humans and robots. Lastly, our detections and recognition results, independently, are extremely competitive to the best detection and recognition results published.

8 NOTATION

This section serves as a reference to the common symbols used in our work.

Symbol	Name	Description
$\Re$	Real Space	space of all real numbers
Γ_1^N		data for training defined as $\{\Gamma_1, \ldots, \Gamma_N\}$
c_1^p	Class Labels	class labels defined as $\{c_1, \ldots, c_p\}$
D	Training Data	defined as $D = \{(\Gamma_n, c_p) : n = 1, \ldots, N; p = 1, \ldots, P\}$
Ω_p	Identity p	is a shorthand symbol for an identity
ψ	Average Face	the average of the training images
ψ_p		the average of the training images for an identity Ω_p
$\mathbf{\Gamma}$		denotes a set of training images ($\mathbf{\Gamma}_l$, $\mathbf{\Gamma}_r$, etc.)
f	Filter	defined as $f(\Gamma) : \Gamma \mapsto \Re^{M'}$
h	Classifier	defined as $h(\Gamma) : \Gamma \mapsto p$, where p is the class label
C	Correlation Matrix	defined as $C = \frac{1}{N} \sum_{n=1}^{N} (\Gamma_n - \psi)(\Gamma_n - \psi)^T$
S	Scatter Matrix	defined as $S = \sum_{n=1}^{N} (\Gamma_n - \psi)(\Gamma_n - \psi)^T$

BIBLIOGRAPHY

[1] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[2] "Robocup@Home," [online]. Available from: `http://www.ai.rug.nl/robocupathome/` [cited 2008-03-26].

[3] T. van der Zant and T. Wisspeintner, "Robocup x: A proposal for a new league where robocup goes real world.," in *RoboCup*, pp. 166–172, 2005.

[4] S. Schiffer, A. Ferrein, and G. Lakemeyer, "Football is coming home," in *Proc. of International Symposium on Practical Cognitive Agents and Robots*, University of Western Australia Press, 2006.

[5] R. Marée, P. Geurts, J. Piater, and L. Wehenkel, "A generic approach for image classification based on decision tree ensembles and local sub-windows," in *Proc. 6th Asian Conference on Computer Vision (ACCV)*, Jan 2004.

[6] R. Marée, P. Geurts, J. Piater, and L. Wehenkel, "Decision trees and random subwindows for object recognition," in *ICML workshop on Machine Learning Techniques for Processing Multimedia Content (MLMM2005)*, 2005.

[7] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 36, no. 1, pp. 3–42, 2006.

[8] P. Geurts, D. deSeny, M. Fillet, M. Meuwis, M. Malaise, M. Merville, and L. Wehenkel, "Proteomic mass spectra classification using decision tree based ensemble methods," *Bioinformatics*, vol. 21, no. 14, pp. 3138–3145, 2005.

[9] R. Marée, P. Geurts, and L. Wehenkel, "Random subwindows and extremely randomized trees for image classification in cell biology," *To appear in BMC Cell Biology supplement on Workshop of Multiscale Biological Imaging, Data Mining and Informatics*, 2007.

[10] R. Marée, P. Geurts, J. Piater, and L. Wehenkel, "Random subwindows for robust image classification.," in *CVPR (1)*, pp. 34–40, 2005.

[11] V. Lepetit, P. Lagger, and P. Fua, "Randomized trees for real-time keypoint recognition," in *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, (Washington, DC, USA), pp. 775–781, IEEE Computer Society, 2005.

[12] F. Moosmann, B. Triggs, and F. Jurie, "Fast discriminative visual codebooks using randomized clustering forests," in *Advances in Neural Information Processing*

Systems 19 (B. Schölkopf, J. Platt, and T. Hoffman, eds.), pp. 985–992, Cambridge, MA: MIT Press, 2007.

[13] P. Yin, A. Criminisi, J. Winn, and I. Essa, "Tree-based classifiers for bilayer video segmentation," in *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, 2007.

[14] T. Deselaers, A. Criminisi, J. Winn, and A. Agarwal, "Incorporating on-demand stereo for real time recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, (Minneapolis, MN, USA), June 2007.

[15] P. Geurts, A. Blanco, and L. Wehenkel, "Segment and combine approach for biological sequence classification," in *Proc. IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB 2005)*, pp. 194–201, 2005.

[16] J. Matas and Š. Obdržálek, "Object recognition methods based on transformation covariant features," in *XII. European Signal Processing Conference EUSIPCO - 2004* (F. Hlawatsch, G. Matz, M. Rupp, and B. Wistawel, eds.), (Vienna, Austria), pp. 1333 – 1336, TU Vienna, Sept 2004.

[17] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[18] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ, USA: Prentice Hall, Inc., 1996.

[19] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, September 1994.

[20] L. Breiman, J. Friedman, R. Olsen, and C. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[21] T. Ho, "Random decision forest," in *Proc. of the 3rd Int'l Conf. on Document Analysis and Recognition*, pp. 278–282, August 1995.

[22] T. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, 1998.

[23] E. Nowak, F. Jurie, and B. Triggs, "Sampling strategies for bag-of-features image classification," in *European Conference on Computer Vision (4)*, vol. 3954 of *Lecture Notes in Computer Science*, pp. 490–503, Springer, 2006.

[24] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother, "Bi-layer segmentation of binocular stereo video," *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 1186 vol. 2–, 20-25 June 2005.

[25] M.-H. Yang, D. J. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 1, pp. 34–58, 2002.

[26] W. Zhao, R. Chellappa, A. Rosenfeld, and P. Phillips, "Face recognition: A literature survey," in *ACM Computing Surveys*, pp. 399–458, 2003.

[27] M. H. Zweig and G. Campbell, "Receiver-operating characteristic (roc) plots: a fundamental evaluation tool in clinical medicine," *Clin. Chem.*, pp. 561–577, 1993.

[28] H. Rowley, S. Baluja, and T. Kanade, "Human face detection in visual scenes," in *Advances in Neural Information Processing Systems* (D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, eds.), vol. 8, pp. 875–881, 1996.

[29] M. Pham and T. Cham, "Fast training and selection of haar features using statistics in boosting-based face detection," *IEEE 11th International Conference on Computer Vision*, pp. 1–7, 2007.

[30] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *CVPR01*, pp. 511–518, 2001.

[31] C. Kotropoulos and I. Pitas, "Rule based face detection in frontal views," *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, April 1997.

[32] G. Yang and T. Huang, "Human face detection in complex background," *Pattern Recognition*, vol. 27, no. 1, pp. 53–63, 1994.

[33] S. A. Sirohey, "Human face segmentation and identification," Master's thesis, University of Maryland, 1993.

[34] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, pp. 679–698, November 1986. Available from: `http://portal.acm.org/citation.cfm?id=11275`.

[35] D. Saxe and R. Foulds, "Toward robust skin identification in video images," in *FG '96: Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition (FG '96)*, (Washington, DC, USA), p. 379, IEEE Computer Society, 1996.

[36] H. Graf, E. Cosatto, D. Gibbon, M. Kocheisen, and E. Petajan, "Multimodal system for locating heads and faces," in *FG '96: Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition*, (Washington, DC, USA), p. 88, IEEE Computer Society, 1996.

[37] B. Scassellati, "Eye finding via face detection for a foveated, active vision system," in *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, (Menlo Park, CA, USA), pp. 969–976, American Association for Artificial Intelligence, 1998.

[38] A. S. Yuille, D. S. Cohen, and P. W. Hallinan, "Feature extraction from faces using deformable templates," *Int. J. Comput. Vision*, vol. 8, no. 2, pp. 99–111, 1992.

[39] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[40] P. Belhumeur, J. Hespanha, and D. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, 1997.

[41] M. Turk, "A random walk through eigenspace(special issue on machine vision applications)," *IEICE transactions on information and systems*, vol. 84, no. 12, pp. 1586–1595, 2001.

[42] K. K. Sung and T. Poggio, "Example-based learning for view-based face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 39–51, 1994.

[43] H. Rowley, S. Baluja, and T. Kanade, "Rotation invariant neural network-based face detection," in *Proceedings of Computer Vision and Pattern Recognition, 1998*, (Washington, DC, USA), p. 963, IEEE Computer Society, 1998.

[44] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. "Hypermedia image processing reference," [online]. Available from: `http://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm` [cited 2008-03-26].

[45] W. K. Pratt, *Digital Image Processing*. John Wiley & Sons, Inc., 2001.

[46] J. Winn, A. Criminisi, and T. Minka, "Object categorization by learned universal visual dictionary.," in *ICCV*, pp. 1800–1807, 2005.

[47] D. Lowe, "Distinctive image features from scale-invariant keypoints," in *International Journal of Computer Vision*, vol. 20, (Hingham, MA, USA), pp. 91–110, Kluwer Academic Publishers, 2004.

[48] P. Yang, S. Shan, W. Gao, S. Li, and D. Zhang, "Face recognition using adaboosted gabor features," in *AFGR04*, pp. 356–361, 2004.

[49] A. Globerson and N. Tishby, "Sufficient dimensionality reduction," *J. Mach. Learn. Res.*, vol. 3, pp. 1307–1331, 2003.

[50] C. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, (Washington, DC, USA), p. 555, IEEE Computer Society, 1998.

[51] B. Manjunath, "Gabor wavelet transform and application to problems in computer vision," in *26th Asilomar Conference on Signals, Systems and Computers*, pp. 796–800, Oct 1992.

[52] B. Moghaddam and A. Pentland, *Beyond Euclidean Eigenspaces: Bayesian Matching for Visual Recognition*. berlin: Springer-Verlag, 1998.

[53] A. Leonardis and H. Bischof, "Dealing with occlusions in the eigenspace approach," in *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*, (San Francisco, CA), pp. 453–458, June 1996.

[54] H. Grabner, P. Roth, and H. Bischof, "Eigenboosting: Combining discriminative and generative information," *IEEE Conference on Computer Vision and Pattern Recognition, 2007*, pp. 1–8, June 2007.

[55] H. Murase and S. Nayar, "Image spotting of 3d objects using parametric eigenspace representation," in *Scandinavian Conference on Image Analysis (SCIA)*, pp. 325–332, June 1995.

[56] A. Leonardis and H. Bischof, "Robust recognition using eigenimages," *Computer Vision and Image Understanding: CVIU*, vol. 78, no. 1, pp. 99–118, 2000.

[57] A. Pentland, B. Moghaddam, and T. Starner, "View-based and modular eigenspaces for face recognition," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'94)*, (Seattle, WA), June 1994.

[58] M. Jones and P. Viola, "Face recognition using boosted local features," in *Proceedings of International Conference on Computer Vision*, April 2003.

[59] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, (New York, NY, USA), pp. 144–152, ACM, 1992.

[60] V. N. Vapnik, *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.

[61] G. Guo, S. Z. Li, and K. Chan, "Face recognition by support vector machines," in *FG '00: Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, (Washington, DC, USA), p. 196, IEEE Computer Society, 2000.

[62] C. Chang and C. Lin, *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[63] C. Hsu, C. Chang, and C. Lin, *A Practical Guide to Support Vector Classification*. Taipei, Taiwan, 2008.

[64] J. N. Darroch and D. Ratcliff, "Generalized iterative scaling for log-linear models," *Annals of Mathematical Statistics*, vol. 43, pp. 1470–1480, 1972.

[65] R. Paredes, "Universidad Politecnica de Valencia Face Database." DSIC/ITI, Universidad Politecnica de Valencia, Spain.

[66] "CBCL Face Recognition Database," [online]. Credit is hereby given to the Massachusetts Institute of Technology and to the Center for Biological and Computational Learning for providing the database of facial images. Available from: `http://www.ai.mit.edu/projects/cbcl` [cited 2008-03-26].

[67] "Yale Face Database," [online]. Available from: `http://cvc.yale.edu/projects/yalefaces/yalefaces.html` [cited 2008-03-26].

[68] "Yale B Face Database," [online]. Available from: `http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html` [cited 2008-03-26].

[69] T. Sim, S. Baker, and M. Bsat, "The cmu pose, illumination, and expression database," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, pp. 1615–1618, 2003.

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

[71] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," *Proceedings of the Second IEEE Workshop on Applications of Computer Vision, 1994.*,, pp. 138–142, 1994.

[72] "The BioID Face Database," [online]. Available from: `http://www.bioid.com/downloads/facedb/index.php` [cited 2008-03-26].

[73] A. Ferrein, G. Lakemeyer, and S. Schiffer, "Allemaniacs@home 2007 team description," tech. rep., RWTH Aachen University, Aachen, 2006.

[74] "CMU Frontal Face Images," [online]. Available from: `http://vasc.ri.cmu.edu//idb/html/face/frontal_images/index.html` [cited 2008-03-26].

[75] "CMU Profile Face Images," [online]. Available from: `http://vasc.ri.cmu.edu//idb/html/face/profile_images/index.html` [cited 2008-03-26].

[76] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, "The feret evaluation methodology for face-recognition algorithms," in *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1090–1104, 2000.

[77] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[78] J. R. Quinlan, "Bagging, boosting, and c4.5," in *AAAI/IAAI, Vol. 1*, pp. 725–730, 1996.

[79] T. Ho, "A data complexity analysis of comparative advantages of decision forest constructors," *Pattern Analysis & Applications*, vol. 5, no. 2, pp. 102–112, 2002.

[80] Y. Amit, D. Geman, and K. Wilder, "Joint induction of shape features and tree classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 11, pp. 1300–1305, 1997.

[81] H. Tao, R. Crabb, and F. Tang, "Non-orthogonal binary subspace and its applications in computer vision," in *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, (Washington, DC, USA), pp. 864–870, IEEE Computer Society, 2005.

[82] E. Simoncelli and H. Farid, "Steerable wedge filters for local orientation analysis," *IEEE Transactions on Image Processing*, vol. 5, no. 9, pp. 1377–1382, 1996. Available from: `www.cs.dartmouth.edu/farid/publications/ip96.html`.

[83] L. Wiskott, J.-M. Fellous, N. Krüger, and C. von der Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 775–779, 1997.

[84] "Open Computer Vision Library," [online]. 2008-03-26. Available from: `http://sourceforge.net/projects/opencvlibrary/`.

[85] "Glossary of terms," *Machine Learning*, vol. 30, no. 2, pp. 271–274, 1998. Available from: `http://dx.doi.org/10.1023/A:1017181826899`.

[86] T. Deselaers, T. Weyand, and H. Ney, "Image retrieval and annotation using maximum entropy," in *CLEF working notes*, (Alicante, Spain), September 2006.

[87] X. Huang, F. Alleva, H. Hon, M. Hwang, and R. Rosenfeld, "The SPHINX-II speech recognition system: an overview," *Computer Speech and Language*, vol. 7, no. 2, pp. 137–148, 1993.

[88] T. Koelsch, "Local features for image classification," Master's thesis, RWTH-Aachen University, Aachen, Germany, 2004.

[89] P. Blunsom, K. Kocik, and J. R. Curran, "Question classification with log-linear models," in *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, NY, USA), pp. 615–616, ACM, 2006.

[90] T. Wu, C. Lin, and R. Weng, "Probability estimates for multi-class classification by pairwise coupling," in *NIPS 03*, 2003.

[91] C. Hsu and C. Lin, "A comparison of methods for multi-class support vector machines," tech. rep., Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2001.

[92] P. W. Hallinan, "Recognizing human eyes," in *Proc. SPIE Vol. 1570, p. 214-226, Geometric Methods in Computer Vision, Baba C. Vemuri; Ed.* (B. C. Vemuri, ed.), vol. 1570 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pp. 214–226, Sept. 1991.

[93] M. H. Yang, D. Roth, and N. Ahuja, "A snow-based face detector," in *NIPS 00*, 2000.

[94] C. A. Waring and X. Liu, "Face detection using spectral histograms and svms," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 35, no. 3, pp. 467–476, June 2005.

[95] A. Martinez and R. Benavente, "The AR face database," Tech. Rep. 24, CVC, June 1998.

[96] A. Georghiades, P. Belhumeur, and D. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *IEEE Trans. Pattern Anal. Mach. Intelligence*, vol. 23, no. 6, pp. 643–660, 2001.

[97] B. Moghaddam and A. Pentland, "Probabilistic visual learning for object detection," in *International Conference on Computer Vision (ICCV'95)*, (Cambridge, USA), pp. 786–793, June 1995.

[98] S. Nayar, S. Nene, and H. Murase, "Real-time 100 object recognition system," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 12, pp. 1186–1198, 1996.

[99] J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 550–554, 1994.

[100] R. Gross, S. Baker, I. Matthews, and T. Kanade, "Face recognition across pose and illumination," in *Handbook of Face Recognition* (S. Z. Li and A. K. Jain, eds.), Springer-Verlag, June 2004.

[101] N. Winters and J. Santos-Victor, "Information sampling for appearence based 3D object recognition and pose estimationased 3d object recognition and pose estimationased 3D object recognition and pose estimation," in *Proceedings of the 2001 Irish Machine Vision and Image Processing Conference*, (Maynooth, Ireland), September 2001.

[102] H. Shao, T. Svoboda, and L. Gool, "ZUBUB - zurich building database for image based recognition," Tech. Rep. 260, Swiss Federal Institute of Technology, 2003.

[103] H. Murase and S. Nayar, "Visual learning and recognition of 3-d objects from appearance," *Int. J. Comput. Vision*, vol. 14, no. 1, pp. 5–24, 1995.

[104] T. Deselaers, H. Müller, P. Clogh, H. Ney, and T. Lehmann, "The clef 2005 automatic medical image annotation task," *Int. J. Comput. Vision*, vol. 74, no. 1, pp. 51–58, 2007.

[105] J. Stone, *Independent Component Analysis*. MIT Press, 2005.

[106] M. Hoque and M. Fairhurst, "A moving window classifier for off-line character recognition," in *Proc. of the Seventh International Workshop on Frontiers in Handwriting Recognition*, pp. 595–600, September 2000.

[107] C. Dance, J. Willamowski, L. Fan, C. Bray, and G. Csurka, "Visual categorization with bags of keypoints," in *ECCV International Workshop on Statistical Learning in Computer Vision*, 2004.

[108] G. J. Edwards, T. F. Cootes, and C. J. Taylor, "Face recognition using active appearance models," in *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume II* (H. Burkhardt and S.-V. Bernd Neumann, eds.), vol. LNCS-Series 1406–1607, (London, UK), pp. 581–595, Springer-Verlag, 1998.

[109] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," in *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume II*, (London, UK), pp. 484–498, Springer-Verlag, 1998.

[110] R. Chellappa, C. L. Wilson, and S. Sirohey, "Human and machine recognition of faces: A survey.," *Proceedings of the IEEE*, vol. 83, pp. 705–740, May 1995.

[111] D. Heath, S. Kasif, and S. Salzberg, "Induction of oblique decision trees," in *IJCAI*, pp. 1002–1007, 1993.

[112] A. Saxena, J. Schulte, and A. Ng, "Depth estimation using monocular and stereo cues," in 20^{th} *International Joint Conference on Artificial Intelligence*, 2007.

[113] Z. Li and X. Tang, "Bayesian face recognition using support vector machine and face clustering," pp. 374–380, 2004.

[114] X. Lu, "Image analysis for face recognition," tech. rep., Dept. of Computer Science & Engineering, Michigan State University, May 2003.

[115] L. Torres, "Is there any hope for face recognition?," in *Proc. of the 5th International Workshop on Image Analysis for Multimedia Interactive Services*, (Lisaboa, Portugal), pp. 21–23, April 2004.

[116] H. Sahbi and N. Boujemaa, "Coarse to fine face detection based on skin color adaption," in *ECCV '02: Proceedings of the International ECCV 2002 Workshop Copenhagen on Biometric Authentication*, (London, UK), pp. 112–120, Springer-Verlag, 2002.

[117] F. Fleuret and D. Geman, "Coarse-to-fine face detection," *International Journal of Computer Vision*, vol. 41, pp. 85–107, June 2000.

[118] B. Wu and R. Nevatia, "Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors," in *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, (Washington, DC, USA), pp. 90–97, IEEE Computer Society, 2005.

[119] B. Wu and R. Nevatia, "Improving part based object detection by unsupervised, online boosting," in *CVPR07*, 2007.

[120] B. Wu and R. Nevatia, "Simultaneous object detection and segmentation by boosting local shape feature based classifier," in *CVPR07*, 2007.

[121] T. Deselaers, D. Keysers, and H. Ney, "Discriminative training for object recognition using image patches," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, (San Diego, CA), pp. 157–162, June 2005.

[122] T. Deselaers, D. Rybach, P. Dreuw, D. Keysers, and H. Ney, "Face-based image retrieval - one step toward object-based image retrieval," in *First International Workshop on Evaluation for Image Retrieval* (H. Müller and A. Hanbury, eds.), (Vienna, Austria), pp. 25–32, September 2005.

[123] P. Dreuw, T. Deselaers, D. Keysers, and H. Ney, "Modeling image variability in appearance-based gesture recognition," in *ECCV 2006 3rd Workshop on Statistical Methods in Multi-Image and Video Processing*, (Graz, Austria), pp. 7–18, May 2006.

[124] M. Zahedi, P. Dreuw, D. Rybach, T. Deselaers, and H. Ney, "Using geometric features to improve continuous appearance-based sign language recognition," in *17th British Machine Vision Conference*, vol. 3, (Edinburgh, UK), pp. 1019–1028, September 2006.

[125] D. Keysers, T. Deselaers, and T. Breuel, "Optimal geometric matching for patch-based object detection," *Electronic Letters on Computer Vision and Image Analysis*, 2007.

[126] E. Xing, A. Ng, M. Jordan, and S. Russell, "Distance metric learning, with application to clustering with side-information," in *Advances in NIPS*, vol. 15, 2003.

[127] K. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in Neural Information Processing Sys-*

tems 18 (Y. Weiss, B. Schölkopf, and J. Platt, eds.), pp. 1473–1480, Cambridge, MA: MIT Press, 2006.

[128] D. Lowe, "Similarity metric learning for a variable-kernel classifier," *Neural Computing*, vol. 7, no. TR-93-43, pp. 72–85, 1995.

[129] E. Nowak and F. Jurie, "Learning visual similarity measures for comparing never seen objects," in *IEEE Conference on Computer Vision & Pattern Recognition*, pp. 1–8, 2007. See also http://lear.inrialpes.fr/people/nowak/.