Real-Time Face Recognition from Surveillance Video

Chapter · February 2011

DOI: 10.1007/978-3-642-17554-1_8

3 authors, including:

Michael Charles Davis
CERN
18 PUBLICATIONS **173** CITATIONS
SEE PROFILE

Stefan Popov
New Bulgarian University
7 PUBLICATIONS **119** CITATIONS
SEE PROFILE

Chapter 1
Real-Time Face Recognition from Surveillance Video

Michael Davis, Stefan Popa, Cristina Surlea

Abstract This chapter describes an experimental system for the recognition of human faces from surveillance video. In surveillance applications, the system must be robust to changes in illumination, scale, pose and expression. The system must also be able to perform detection and recognition rapidly in real time.

Our system detects faces using the Viola-Jones face detector, then extracts local features to build a shape-based feature vector. The feature vector is constructed from ratios of lengths and differences in tangents of angles, so as to be robust to changes in scale and rotations in-plane and out-of-plane. Consideration was given to improving the performance and accuracy of both the detection and recognition steps.

1.1 Introduction

CCTV cameras have become ubiquitous, nowhere more so than in the United Kingdom. So far, the value of CCTV surveillance has failed to live up to the hype. A recent report by the Metropolitan Police in London revealed that the city's one million-plus cameras have helped to solve only a handful of crimes [18]. The criminologist Clive Norris points out [21] that CCTV operators tend to act on their prejudices (for example, focussing cameras on people because of their skin colour) or merely on scenes which they find entertaining, to relieve the boredom of staring at mundane street scenes all day. So far, the main CCTV success stories are for forensic use[1] and as a deterrent against some forms of petty or opportunistic crime[2].

CCTV surveillance is often justified on the basis that it can be used to prevent terrorist attacks, but the reality is somewhat more banal. As [26] points out, those committing acts of terrorism are usually unknown before the act takes place, and

[1] See [25] for a recent example.

[2] As predicted by Jeremy Bentham's *Panopticon* [4], people change their behaviour when they know they are under surveillance. This is not only true of criminals. For example, ordinary people change their behaviour within airports to flow through the system of control [22].

it is very difficult to conduct surveillance of someone whose identity is unknown. There are few examples of CCTV surveillance being used effectively for real-time detection or prevention of crime.

It has been suggested that the effectiveness of CCTV surveillance can be improved by tracking known individuals as they move through a surveillance network. One possible application could be to follow known hooligans as they enter and move about a football stadium. This paper will consider the design of a video surveillance network which could track the movement of known individuals in real time.

1.1.1 Real-Time Video Surveillance

Let us consider a specific application area — video surveillance within a Secure Corridor; for example, within an airport or other secure building. Airports can be considered as a self-contained microcosm, which can be viewed in terms of the flows through them, and the movements and behaviours that take place in their arrival, shopping and transit areas [15].

The airport can also be viewed as a *Surveillant Assemblage*. The relationship between surveillance and mobility within an airport is considered in [22]. Until recently, airport surveillance did not focus on individuals – people were disassembled into data flows to monitor movement through the airport. Individual surveillance has come to the fore within the last ten years. Passenger profiling is conducted ostensibly for security, but profiling information is also of commercial use to the airport authorities. The combination of profiling with the ability to track the movement of individuals is potent.

A number of airports have already installed systems to register passengers as they enter the Secure Corridor, and track when they leave. Passengers are enrolled using their boarding card and a digital photograph is taken. When the passenger leaves the Secure Corridor, the photograph is compared with the passenger's face by a security officer. This generates a transaction when the passenger enters and leaves the secure area, but cannot track their movements within it. Airports may wish to track some individual passengers as they move through passport control, the retail shopping area, departure lounges and exit to their gate. Our hypothetical system could achieve this by tracking the passenger's face as they move around the secure area.

The proposed system will need to work in real time: that is, the extraction of facial features and comparison to the stored model for each passenger must be extremely fast. The metric used must be reasonably robust to changes in scale, illumination, pose and occlusion. The system will need to track individuals as they move from the field of view of one camera to another. A distributed network of cameras will need to communicate with a back-end database and pattern-matching system. These requirements are discussed in more detail in section 1.2.

1.1.2 Face Recognition

The dream of tracking individuals by their facial characteristics goes back as far as the 1870s. [20] records how a police records clerk in Paris, Alphonse Bertillon, created a system for taking facial metrics from photographs. Distances between facial features were measured and encoded. Police photographs could be filed according to the codes, which reduced the search space for a suspect — police officers had only to compare suspects with photos which had similar measurements, rather than the entire portfolio.

A hundred years after Bertillon, research began in earnest to automate the recognition of faces by machine. By the end of the 20th century, there had been significant progress. [35] surveys the main algorithms in use, as well as considering the neuroscience and cognitive aspects of human facial recognition, and problems such as variation in pose and illumination.

1.1.2.1 Face Recognition by Humans

[29] presents 19 observations on the human ability to recognise faces, all of which are relevant to research in computer vision. The results most relevant to the proposed face tracking system are mentioned below.

It is clear that humans adopt a holistic approach to identifying one another. Recognition may be multi-modal (for example, combining face data with body shape or gait), but the studies in [29] indicate that facial information is one of the principal ways in which we recognise each other. A multi-modal recognition system could take metrics such as gait into consideration, combined with facial recognition to increase robustness.

Recognition is not only multi-modal but also multi-method. Humans combine multiple visual cues to recognise a face. In designing machine algorithms for face recognition, we acknowledge that any technique is only part of the solution. A robust recognition system will combine results from multiple techniques.

[29] asserts that the spatial relationship between facial features is more important than the shape of the features themselves. Like humans, machines can accurately recognise faces from a low-resolution image (see [27]) as these spatial relationships are preserved. Skin pigmentation cues are at least as important as shape cues and these are also preserved at low resolution.

Besides the role of pigmentation in recognition, colour is important for segmentation. Especially at low resolutions (where shape information may be degraded), colour information allows humans to estimate the boundaries of image features much more accurately than when presented with a greyscale photo. Likewise, colour cues can help a computer to locate local facial features such as the eyes. Analysis of the hue histogram of an image allows better estimation of the shape and size of the eyes than the luminance histogram alone.

In considering local features, consideration is usually given to the eyes, nose, mouth and possibly the ears. However, the studies in [29] revealed that the eyebrows

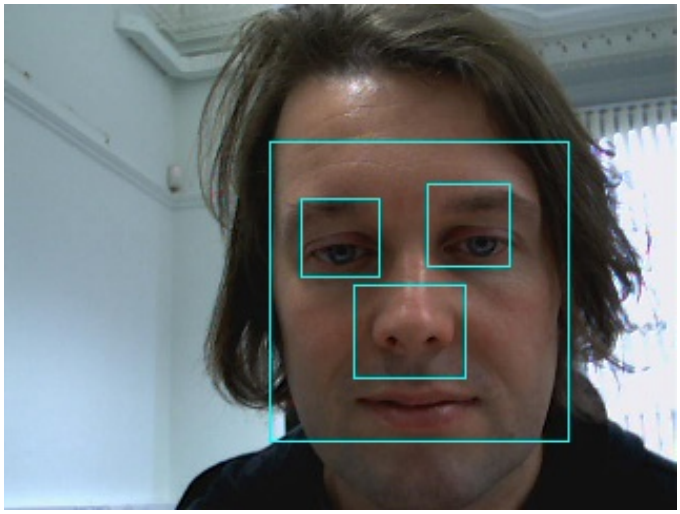

Fig. 1.1 OPENCV face and local feature detection using nested Haar cascades

are one of the most important local features in human facial recognition. Eyebrows have several features which make them suitable for machine recognition: they are stable, with high contrast, which makes them easy to acquire even in a low resolution image. Also, they sit on a convex part of the face, which makes them less susceptible to shadow and illumination problems.

It is interesting to note that faces can be stretched or compressed without any loss of recognisability. Humans may be able to adapt to such transformations because they are similar to the effect of a facial rotation out-of-plane. It is notable that ratios of distances between features in the same dimension remain constant under stretching/compression transformations. Humans may use such *iso-dimension ratios* for recognition. This suggests that machine facial recognition algorithms could adopt the same approach. Instead of the usual approach of pose estimation (see discussion below), the facial metric could be based on ratios which are invariant to changes in scale and rotation. This idea will be developed further in section 1.4.1.

Finally, we note that humans make temporal associations when observing the movements of someone else's face. This helps to build a more general model of the face.

1.1.2.2 Face Recognition by Machine

The most important face recognition algorithms and techniques are discussed in [35] and [34].

Holistic, or **appearance-based**, approaches to face recognition were pioneered by Turk *et. al* in their paper on Eigenfaces [30], where differences in human faces are represented by a set of eigenvectors. A covariance matrix is calculated over a set

of training images for the face of an individual. The eigenvectors and eigenvalues of the covariance matrix are extracted using Principal Component Analysis (PCA). This has the effect of projecting the high dimensionality of the face image space into a feature space of lower dimensionality. Features are classified on a nearest-neighbour basis. The technique can be applied to local features as well as whole faces (in this case the Principal Components are referred to as Eigeneyes, Eigennose, *etc.*) Eigenfaces produces very reliable results under laboratory conditions, but is not robust to changes in pose, illumination and expression. It also has high computational requirements, making it unsuitable for real-time facial recognition from a surveillance video stream.

Turk discusses improvements to and extensions of his Eigenfaces technique in [34]. PCA builds its model by minimising the pairwise relationships between pixels in the image training set. The approach can be generalised to minimise the second-order and higher-order dependencies in the image space, using Independent Component Analysis (ICA). ICA attempts to find the basis along which the data are statistically independent. It was first applied to face recognition in [2], where ICA was shown to be more robust than Eigenfaces to minor changes in illumination, expression and appearance (hair, make-up).

Another well-known appearance-based approach is Fisherfaces [3]. The Fisherface approach attempts to improve on Eigenfaces by creating a model which is invariant to changes in illumination and expression (but not pose). Fisherfaces uses Linear Discriminant Analysis (LDA), which is similar to PCA in that the high dimensionality of the image space is reduced to a lower-dimensional feature space. Fisherfaces attempts to choose the direction of the projection such that variations in lighting and facial expression are projected away but the features used for recognition are clustered. PCA chooses the projection which maximises the total scatter, preserving unwanted variations in lighting and facial expression. The computational requirements of the two approaches are similar. Fisherfaces does not solve the problem of variations in pose. The image in the test set must be similar to the image in the training set to be detected. The test set must contain a comprehensive set of images under different lighting conditions.

PCA, LDA and ICA are compared in [28]. Another similar approach which deserves a mention is Tensorfaces [1]. Tensorfaces uses multi-linear analysis, a variation of PCA, to combine training sets of face images under different poses, expressions and lighting.

All the holistic approaches require a sizeable training database of face images under different poses, expressions and lighting to provide accurate results. For a surveillance application, it is unlikely that an accurate model of all possible facial variations can be constructed in advance. The computational requirements are also considerable. These challenges mean that holistic, appearance-based methods have not been demonstrated to work accurately in real-world situations. Turk concludes [34] that Eigenface or other appearance-based approaches must be combined with feature- or shape-based approaches to recognition to achieve robust systems that will work in real-world environments.

Structural, or **feature-based**, approaches to face recognition begin by locating local features such as the eyes, nose and mouth. The locations and characteristics of the features are then used to classify the face. As discussed in [35] and [34], early feature-based approaches were based on pure geometry methods, which match on measurements between features, such as the distance between the eyes or from the eyes to the mouth. A more sophisticated approach is Elastic Bunch Graph-Matching [32], which represents faces using labelled graphs, based on a Gabor wavelet transform. Image graphs of faces are compared with a similarity function. The technique was shown to be robust to small changes in rotation (up to 22°). One advantage of this method is that it requires only a single photo to create the elastic bunch graph.

Some recent developments have attempted to synthesise appearance-based and feature-based methods in a hybrid approach. [36] uses semantic features (the eyes and mouth) to define a triangular facial region. Tensor subspace analysis is applied only to this region to create the feature vector. This has the advantage of combining the geometric information from the spatial locations of the local features and the appearance information from the texture of the facial region. The computational requirements are much less than Eigenfaces or Tensorfaces alone, and the accuracy is higher. However, the problems of variations in illumination, pose and scale are not considered.

1.1.2.3 Face Recognition from a Video Stream

The machine recognition techniques considered so far were considered in the context of recognising faces from still images. For a surveillance application, we must also consider the challenges (and opportunities) of recognising faces from a video stream. In a surveillance video, we can expect that faces will be captured under variable scale, pose and illumination, and that occlusions will be likely. In our proposed scenario (a Secure Corridor in an airport), we can assume that the environmental lighting is constant, but there will still be variation in terms of lighting direction and shadow.

Almost all of the techniques considered so far assume that it is possible to capture a full-frontal view of the face, but we cannot make this assumption in a surveillance application. Pose angle can significantly decrease recognition accuracy. [13] tackles this problem through a pose-estimation step. Faces are extracted from video frames using an Adaboost classifier, then the same technique is used to locate local features (eyes) within the face area. The location of the centre-point between the eyes relative to the centre-point of the detected face area is used to assign a Gaussian weight to each captured face frame. Face frames which are closer to normal (frontal) pose will be given a higher weight during the recognition step.

The problem of occlusions is considered in [17]. Part of the face under consideration may be occluded, for example by long hair, sunglasses, a scarf or dark shadow. The approach taken is “recognition by parts”, where the facial image is broken up into sub-blocks which can be considered individually. Features based on wavelet co-

efficients are extracted from each sub-block. It is assumed that some of the features are corrupt, but it is not known in advance which ones. A Posterior Union Model is used to find the set of features which maximise the probability of a correct match.

We should also recognise that the video modality gives us certain advantages compared to face recognition from static images. As mentioned in [29], humans observe temporal relationships when recognising someone's face. Our machine facial recognition system can also exploit the temporal relationship between video frames. For example, well-established tracking methods such as particle filtering [11] could be used to track the movement of a face within a video stream. Multiple samples of the same face can then be extracted and passed to the recognition system. As we can take many samples, we can weigh or discard samples which are of poor quality.

Video modality can also be combined with recognition from other modalities using multi-modal fusion algorithms. [13] combines facial recognition with voice recognition to improve accuracy. Gait analysis could also be considered, as it would not require any additional sensors.

1.1.3 Real-Time Face Recognition from Surveillance Video

We have considered the state-of-the-art in face recognition technology. Face recognition in laboratory conditions is quite mature, but there are many outstanding problems which prevent accurate real-time recognition of faces from a video stream. Our experimental system attempts to investigate some of these problems.

As discussed above, holistic (appearance-based) approaches are not suitable for surveillance applications. They are sensitive to facial expressions and occlusions, and face images must be accurately normalised for pose, illumination and scale. Eigenfaces and similar approaches also have a feature vector of high dimensionality, which makes the process of matching computationally expensive. We have therefore adopted a shape-based method. [9] proposes a shape-based method for recognising faces from a video stream using a combination of a fixed (CCD) and PTZ camera. We have calculated a feature vector following a novel approach suggested by [33].

The recognition system must be robust to changes in illumination, scale, expression, pose and occlusion:

- **Illumination**: detecting local features within a face is reasonably robust to changes in illumination.
- **Scale**: the feature vector is constructed from ratios of lengths and angles, so is invariant to changes in scale.
- **Expression**: we have selected features from the non-deformable parts of the face to provide invariance to changes in expression.
- **Pose**: we do not assume that we will have a frontal view of the subject. Two models are constructed of each subject, one frontal view and one profile view. The main part of the feature vector is constructed from differences in ratios of

angles, which provides reasonable robustness to rotation within one view. A pose estimation step is therefore not required.
- **Occlusion**: as we only need to sample features from the non-deformable part of the face, and we extract multiple samples from the video stream, there is reasonable robustness to occlusions.

The feature vector is extracted from the spatial relationships between local features. It is therefore of very small dimensionality, allowing fast transmission over the network and rapid matching to the database.

Our system was developed and tested using the following platforms and test data:

- Face detection was implemented in C using the OPENCV [5] libraries.
- Feature extraction algorithms were developed using MATLAB®. The test data included a live webcam stream, digital photos taken by the authors and the FERET database of face images.
- Feature extraction was then ported to C++/OPENCV. Test data included a live webcam stream and videos of pose variation from CSIT[3].
- The classification and matching system was implemented in C++ using the OPENCV SVM class.

Section 1.2 describes the design and architecture of our experimental system. Sections 1.3 and 1.4 describe the algorithms and operation of the front-end and back-end, respectively. Our experimental results and conclusions are discussed in sections 1.5 and 1.6.

1.2 System Architecture

Our experimental system has been designed with the requirements of a real-time, distributed sensor network in mind. It is envisaged that sensor nodes would be located all around the Secure Corridor. Each node would contain a video camera, an on-board processor and a network connection. The nodes would detect faces, then send data to a back-end for matching to the database. The system architecture is shown in figure 1.2.

First, faces are detected by the distributed sensor nodes. Feature extraction is also carried out by the node. The high dimensionality of each video frame is reduced to a feature vector of very low dimensionality. This vastly reduces the network bandwidth required and spreads the processing load across the distributed network. Object tracking could be carried out by the nodes, so that multiple face samples are attributed to the same person. Feature vectors can then be tagged with an identifier and sent to the back-end. (Object tracking was not implemented in our experimental system).

[3] Centre for Secure Information Technologies, Queen's University, Belfast (http://www.csit.qub.ac.uk/). Thanks to Dr. Darryl Stewart and Adrian Pass for providing the video database [24].

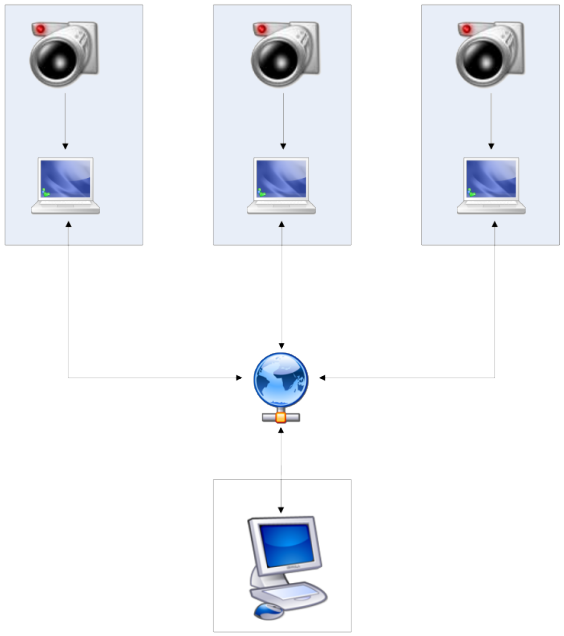

Fig. 1.2 Distributed System Architecture: Multiple Front-Ends

The back-end system receives feature vector packets and unpacks them for pattern matching with the database. This is treated as a classification problem, with each individual enrolled on the system representing a class. Matching is performed using a Support Vector Machine (SVM). Temporal relationships between frames could be exploited to increase the likelihood of a correct match. In a production systen, the back-end would log the time and location (camera no.) of individuals under surveillance, and would be configured to send alarms to operators.

Please see the Appendix to this chapter for more implementation details, including how to obtain the source code.

1.3 Front-End

The front-end has three functions: to detect faces, to track detected faces and to extract features for transmission to the back-end recognition system. The process is summarised in figure 1.3.

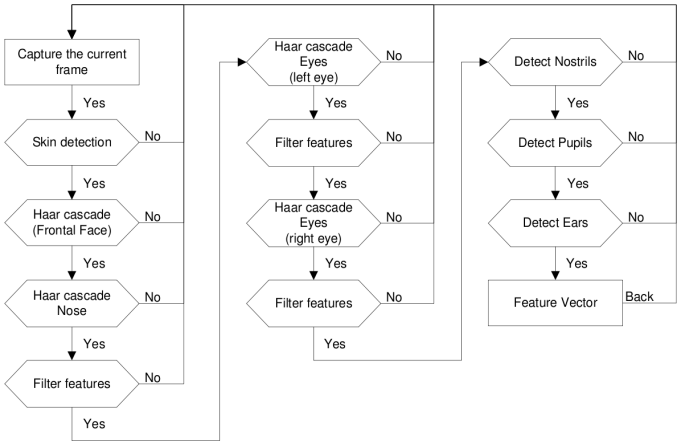

Fig. 1.3 Front End: Face Detection and Feature Extraction

1.3.1 Face Detection

Face Detection is carried out using the OPENCV Face Detector [5]. This is an implementation of the Viola-Jones Face Detector [31], which uses a boosted rejection cascade based on Adaboost [8].

The theory of boosting is that many weak classifiers can be combined to make a strong classifier. Adaboost is a supervised classifier, *i.e.*, it takes a set of labelled samples as input. A tree of weak classifiers is created by applying a weak algorithm to the labelled data. Each weak classifier is assigned a weight based on its performance on the training data. Classifiers that return the wrong result are given a higher weight, to concentrate attention on the places where errors are likely to be made. If this process is applied iteratively, the training error of the strong classifier exponentially approaches zero. This is repeated until the error rate drops below a set threshold.

Viola-Jones uses Adaboost both for feature selection and to train the face detector. During feature selection, Adaboost can select a few important features from the set of all possible features. The features are Haar-like wavelets. The Haar-like features are extracted by applying a threshold to the sums and differences of rectangular image regions. OPENCV also includes diagonal wavelets[4] to detect rectangular features rotated 45°. Each weak classifier in OPENCV typically uses one feature (at most three).

A weak classifier $h(x, f, p, \theta)$ is used, where x is a 24×24 pixel sub-window of an image, f is a feature, θ is a threshold and p is the polarity (direction) of the inequality:

$$h(x, f, p, \theta) = \begin{cases} 1 \text{ if } pf(x) < p\theta \\ 0 \text{ otherwise} \end{cases}$$

[4] This extension to the Viola-Jones technique is described in [16]

Each stage of the boosting process attempts to find the Haar-like feature which best separates faces from non-faces.

One of Viola-Jones' innovations was the use of integral images to rapidly compute the Haar-like features at any scale or location in the image. The integral image at (x,y) is defined as the sum of all pixels above and to the left of (x,y). The sum of all pixels within any given rectangle can be rapidly calculated from the sums and differences of the four corner pixels of the integral image. By thresholding these rectangles, the Haar-like features can be calculated at any location or scale without needing to build a multi-scale pyramid.

Another important innovation was the use of a cascade of classifiers to reduce the computation time and increase accuracy. First, a two-feature strong classifier is applied to the integral image. The threshold is set to minimise false negatives at the cost of a high false positive rate (up to 50%). This rapidly rejects background regions and focuses attention on more promising regions. Regions which pass the first classifier are passed to the next classifier in the cascade. The strategy is to reject negatives as early as possible and gradually reduce the number of false positives as the regions pass through the cascade. A region that passes through every level of the cascade is classified as a face.

The Viola-Jones method can be applied to other objects besides faces, but works best on objects with "blocky" features rather than those where the outline is the most distinguishing characteristic. This is because the classifier must include part of the background of an object in its model of the object's outline.

OpenCV includes pre-trained cascades for frontal faces, profile faces, eyes, noses and mouths. Our implementation used nested cascades within the OpenCV face detector to find local features (eyes and noses) within detected faces (see figure 1.4). Feature extraction is discussed in section 1.3.3.

The detection of profile views works well when the background is plain but less well with a textured background, for the reason mentioned above. The detection of profile faces was improved using skin detection (see below).

1.3.1.1 Skin Detection

Viola-Jones operates on greyscale images, but for our implementation, we decided to also exploit the colour information in the video stream[5]. By applying skin colour detection to the input video frames, it is possible to divide the search space into face candidate regions and background regions. Face detection need only be applied to face candidate regions. The segmentation of skin regions based on luminance and chrominance values is discussed in [10].

OpenCV's Adaptive Skin Detector was used to detect face candidates (see figure 1.5). Skin-coloured areas (with a margin added around the edges to allow for areas of shadow) were passed to the face detector. This resulted in a significant

[5] Indeed, the original Viola-Jones paper [31] suggests that alternative sources of information can be integrated with the basic approach, to achieve even faster processing and better frame rates.

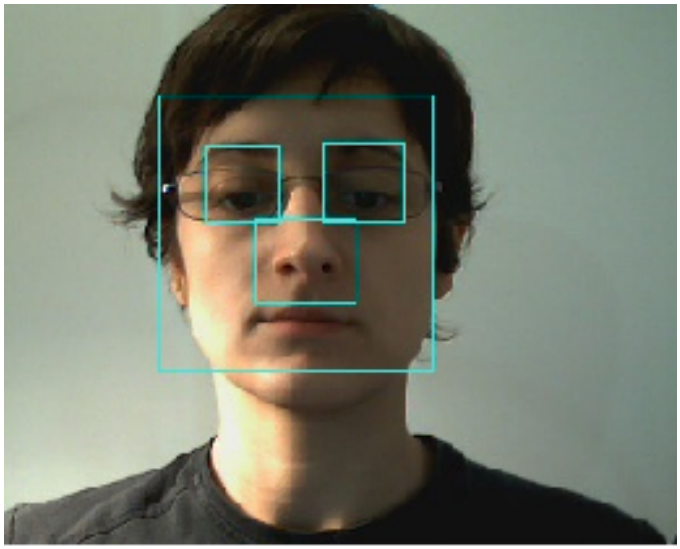

Fig. 1.4 OPENCV face and local feature detection

speedup and also improved accuracy, by discarding face-like regions which are not skin-coloured (false positives).

Another benefit of using the skin detector was to segment faces in profile view from the background. This enabled more accurate detection of the facial profile, overcoming the problem of detecting an outline against a textured background (see above).

The results of skin detection are discussed in detail in section 1.5.1.

1.3.2 Face Tracking

As we are examining video sequences rather than still images, we can exploit the temporal relationships between frames to improve recognition accuracy. It is possible to track the movement of individual faces in a video stream using techniques such as Boosted Particle Filtering, Mean Shift, or Kalman Filtering[6].

If we can successfully track a moving face through a video sequence, we can take each frame as being a sample of the same person. Extracted features can be tagged with an identifier before sending to the back-end for recognition. Features with the same identifier can be considered as multi-samples of the same individual.

[6] Linear dynamic models are discussed in [7]. Non-linear models are also discussed in the unpublished chapter, "Tracking with non-linear dynamic models", available at http://luthuli.cs.uiuc.edu/ daf/book/bookpages/pdf/particles.pdf. See also [11].

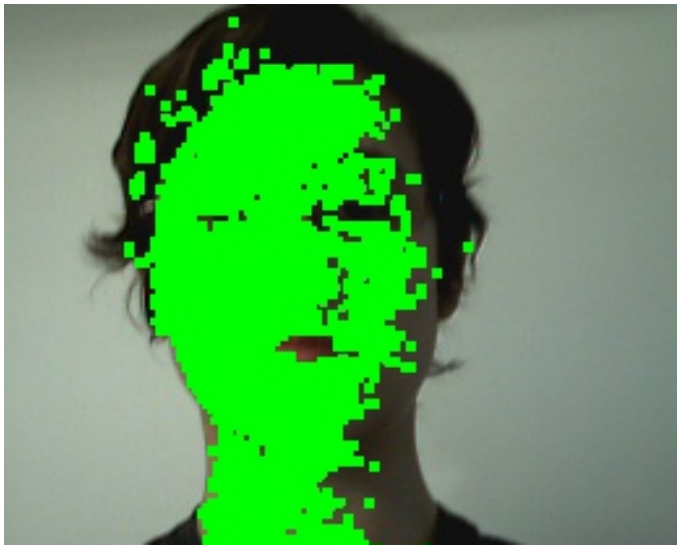

Fig. 1.5 Skin Detection

Note: Multi-sampling was not implemented in the experimental system described in this chapter.

1.3.3 Feature Extraction

One of the objectives of the face recognition system is invariance to changes in pose. In previous approaches, this problem has been addressed by either creating a 3D model [1] or by employing a pose estimation step [13, 32]. The creation of a 3D model is problematic, as it requires a large set of training images for each individual. This is impractical in a real-time surveillance application. Our system follows a novel method, proposed in [33], where the feature vector is invariant to pose, so a pose estimation step is not required.

[31] notes that the face detector can detect faces which are tilted up to about $\pm 15^\circ$ in plane and $\pm 45^\circ$ out-of-plane (towards a profile view). In order to recognise faces from any angle, our system constructs two separate models of each individual: a frontal model and a profile model.

Each model (frontal and profile) defines a **baseline**, a **reference length** and a set of **feature key points**. The feature vector is defined by the spatial relationships between these features (see section 1.4.1).

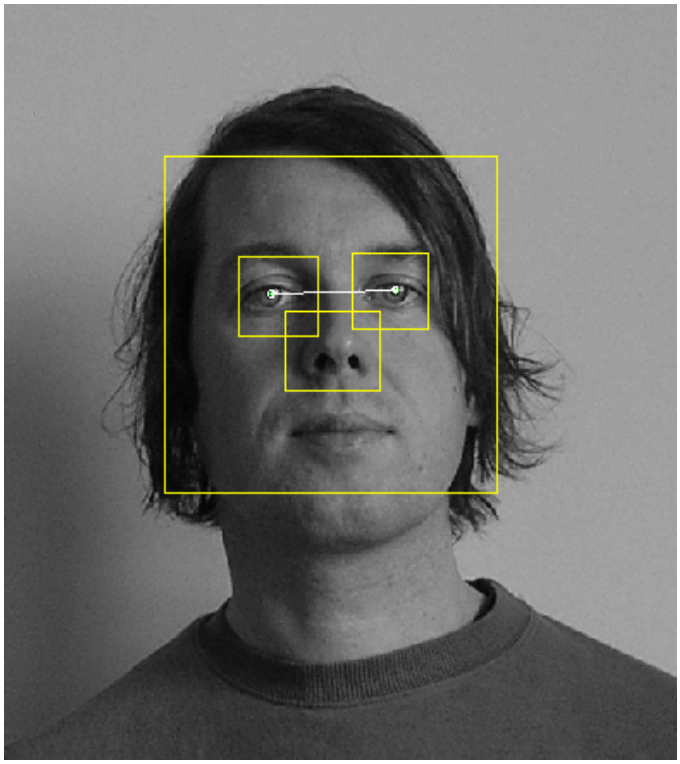

Fig. 1.6 Baseline for frontal face model

1.3.3.1 Frontal Face Model

For the front face model, we first define two reference points — the centre points of the irises. The front baseline is defined as a line connecting the reference points (see figure 1.6. This length of this line is used as the reference length for the front face. Next, we define a number of key points which denote the location of shape-based features such as the nostrils and the tips of the ears. The feature vector is calculated from a set of lengths and angles measured from the baseline to the feature key points.

The outline algorithm for finding the baseline and key feature points is as follows (see figure 1.3):

1. Find the face bounding box using the OpenCV Haar cascade detector
2. Use nested Haar cascades to detect local features (eyes and nose)
3. If local features are not detected, discard the frame
4. Search within eye bounding boxes for reference points
5. Search within nose bounding box for feature key points
6. Search for other key points (for example around face contours)

Note that the reference points and feature key points are chosen from features in the non-deformable part of the face, to provide robustness to changes in facial expression. The detailed algorithms for extracting the reference points and other key points are described in the following sections.

1.3.3.2 Iris Detection

The reference points for the front face are the centres of the two irises. During preprocessing, the eyes have been located and the system has determined the bounding box for two eye regions. Face candidates which do not have two eye regions have been discarded. If more than two eyes have been detected, the system attempts to estimate the most likely eyes by considering their location and size relative to each other and the face bounding box.

The reference points are located at the centre of the iris/pupil within the two eye bounding boxes. There are a number of possible approaches to iris detection. If sufficient training data were available, it would be possible to add a third nested Haar cascade to the OPENCV face detector to detect irises within the eye region. [6] proposes an integrodifferential operator which can detect iris and pupil boundaries. Another possibility would be to match curves to the eye region using a Hough Transform. Circles of approximately the right size could be extracted as irises and pupils. [14] takes this approach, improving robustness by using a separability filter.

Our approach was a geometric approach using thresholding and blob detection on the image. This has the advantage of being simple and fast to compute.

Detection uses the following algorithm[7]:

1. Use a 5×5 averaging filter to remove noise and improve blob detection[8].
2. As the surface of the eye is highly reflective, there is usually a point of light on or near the pupil/iris. To locate this point, a gradient is calculated over the image. Small gradients are discarded. Large gradients are weighted by the distance from the centre of the bounding box. The highest weighted gradient is taken as a point assumed to be on or close to the pupil/iris (see figure 1.7).
3. Histogram equalise to improve blob detection.
4. Threshold image at $0.25 \times \max(\mathit{image})$.
5. Erode thresholded image with a 3×3 structuring element (see figure 1.8)
6. Find the blob closest to the gradient point.
7. Use region selection to find the maximum and minimum points within the iris blob.
8. Calculate the centre of the blob as the average of the maxima/minima. (figure 1.9)
9. **return** centre of blob

[7] Detection algorithms were developed experimentally in MATLAB (see `detect_iris.m`) and subsequently implemented in C++/OPENCV.

[8] Later experimentation with a larger test set showed that the averaging filter did not improve detection accuracy, so this step was removed from the C++ implementation

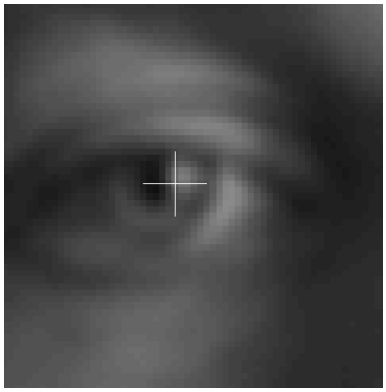

Fig. 1.7 Eye region: blurring filter and location of highest weighted gradient

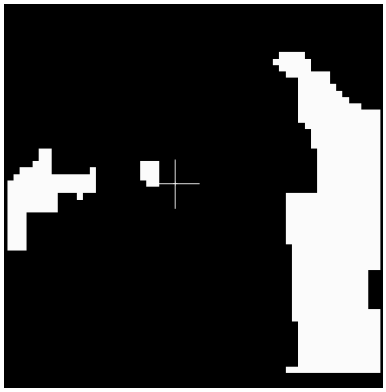

Fig. 1.8 Eye region: thresholding and blob detection

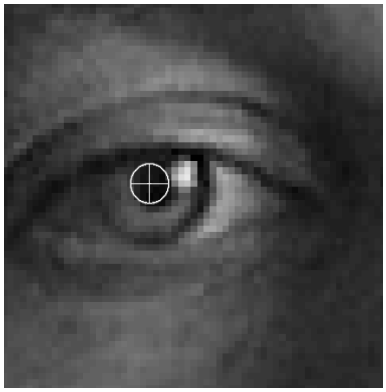

Fig. 1.9 Eye region: detection of centre of pupil/iris

Note that this method works reasonably well on people who are wearing glasses. [3] solves the problem of wearing glasses by dividing images of an individual into two classes, one which is "wearing glasses" and one "not wearing glasses". The OPENCV face detector includes eyes wearing glasses in its cascades for face and eye detection.

Once the two reference points have been determined, the baseline is calculated as the line joining the two points (see figure 1.6).

1.3.3.3 Nostril Detection

Having detected the two reference points, the nostrils are detected next, as the first feature key points. The nostrils are important, as they are rarely occluded and establish the location of the centre of the face. The location of the nostrils is used to calculate a centre-of-face reference line which is perpendicular to the baseline.

Nostril detection is similar to iris detection. It would be possible to use a third nested Haar cascade if we had a training set of nostril shapes. Again, it would be possible to use a Hough transform to detect the roughly-elliptical nostril shapes. We adopted a geometric approach similar to the method used for iris detection. This works very well, as the nostrils appear clearly as two dark blobs in the thresholded image. The detailed algorithm is as follows:

1. Use a 5×5 averaging filter to remove noise and improve blob detection[9].
2. A gradient is calculated over the image and used to estimate the direction of illumination. Typically, illumination is from one side, meaning that one side of the nose is illuminated and the other side is in shadow. The highest gradient on the blurred image is shown in figure 1.10.
3. Local histogram equalisation is performed independently on both sides of the nose (see figure 1.11). This compensates for unequal lighting.
4. Threshold image at $0.25 \times \max(image)$.
5. Erode thresholded image with a 3×3 structuring element (see figure 1.12)
6. Find the two blobs closest to the estimated tip of the nose[10].
7. Use region selection to find the maximum and minimum points within the nostril blobs (see figure 1.13).
8. **return** nostril boundary points

Once the nostril points have been detected, it is possible to calculate the centre face reference line. A point is calculated as the midpoint of the two nostrils, and a line perpendicular to the baseline is drawn through this point (see figure 1.14). The point where the baseline and centre face line intersect is used as the reference point for key feature measurements.

---

[9] Later experimentation with a larger test set showed that the averaging filter did not improve detection accuracy, so this step was removed from the C++ implementation

[10] Initially our algorithm was thrown off by reflected light from Adrian's nose-ring but we were able to compensate for this!

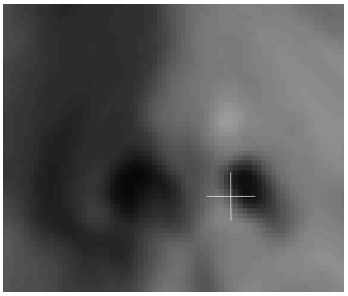

Fig. 1.10 Nose region: blurring filter and location of highest weighted gradient

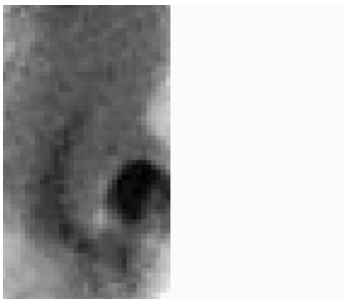

Fig. 1.11 Nose region: local histogram equalisation

Fig. 1.12 Nose region: thresholding and blob detection

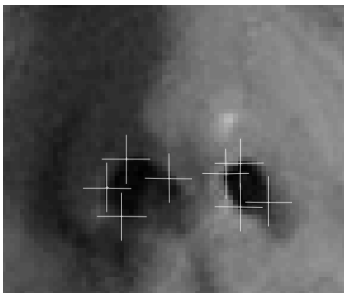

Fig. 1.13 Nose region: detection of boundary points

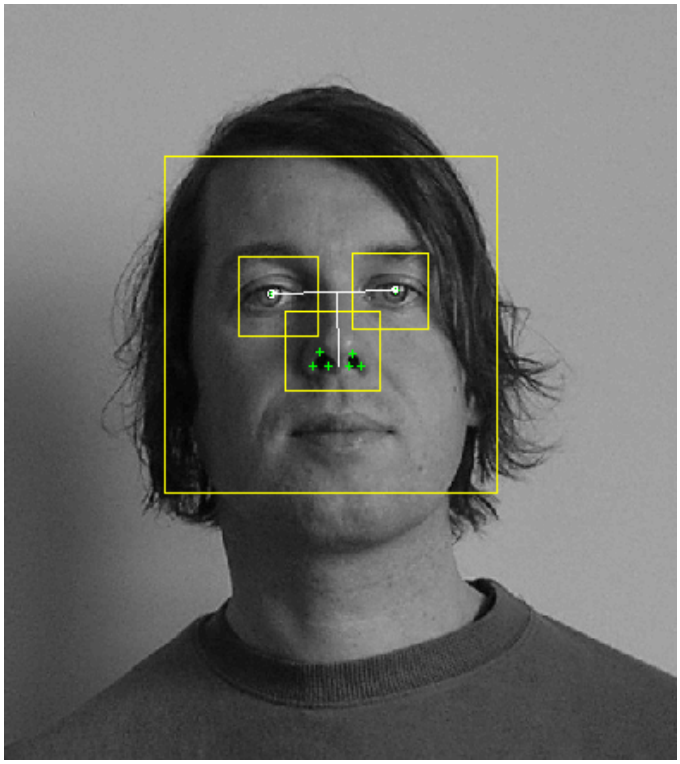

Fig. 1.14 Front Face Model, showing Baseline, nostril feature key points and centre reference line

1.3.3.4 Ear-Tip Detection

The third set of features that we use for the facial feature vector are ear-tips. Unlike irises and nostrils, which can be detected as blobs on the front of the face, ear-tips can be detected as a corner along the edge of the face.

If a suitable set of training data was available, it would be possible to detect ears using a nested Haar cascade as we do for eyes and nose. However, this introduces the same problems as profile detection, in that it is difficult to train a Haar cascade to detect edges reliably in the presence of textured backgrounds. This could be compensated for using skin detection as previously discussed.

As we have already located the eyes, we use this as a cue to detect the location of the ears. As the Haar cascade face detector sometimes returns a bounding box which has cropped the ears, we expand the detection area to compensate for this.

The algorithm that we used for ear-tip detection is as follows:

1. Detect within the region returned from the OPENCV face detector.

2. If the skin patch representing this face is wider than the bounding box returned by the Haar cascade, expand the detection region as required.
3. Use OPENCV morphological operations to smooth the image and remove internal edges. This leaves the outside contour of the face as the only strong edge.
4. Perform Canny edge detection to get the face outline (see figures 1.16 and 1.17).
5. Detect corners by calculating gradients along the face edges
6. Choose the lowest sharp corner on either lateral side of the face. This should be located where the ear lobe meets the face outline.
7. **return** ear tip points

Ear detection is much less reliable than detection of local features in the front-of-face, as we cannot guarantee that the ear will not be occluded. Even if the ears are clearly visible, our algorithm may fail if the jaw-line is within the region of interest, as this will result in another sharp corner lower than the ear.

Detected ear-tips are shown in figure 1.15.

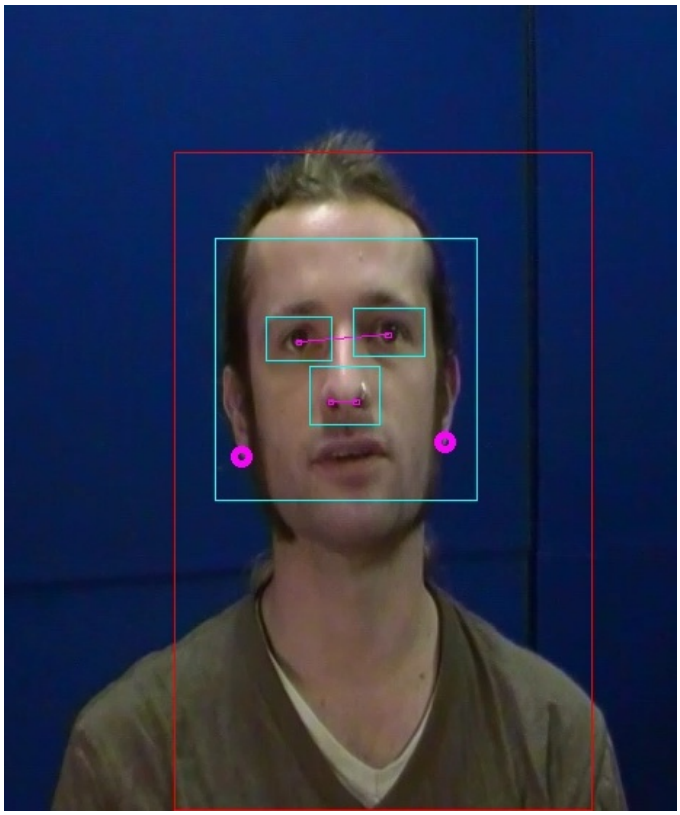

Fig. 1.15 Ear-tip detection

Fig. 1.16 Ear outline detection: Darryl

Fig. 1.17 Ear outline detection: Adrian

Fig. 1.18 Baseline for profile face model

1.3.3.5 Profile Face Model

The model for the profile view is constructed using the same principles as the frontal face. The image is searched using the OpenCV face detector with the cascade for detecting profile faces. When a face is detected, a nested Haar cascade is used to detect an eye (naturally, only one eye is visible in profile view).

The baseline for profile view is again constructed using two reference points. The first point is at the top of the nose. The second point is located by fitting a line to the profile of the nose, and then finding a perpendicular line which is tangential to the bottom of the nose. The point where this line meets the upper lip is the second reference point. The baseline is constructed by joining these two points. The length of the baseline is the reference length for the profile view. The profile baseline is shown in figure 1.18.

OpenCV provides a Haar cascade only for left-facing profiles. Normalising all training samples to the same view is a more efficient use of the data. In order to detect right-facing profiles, the images must first be horizontally flipped.

As previously discussed, using the OPENCV face detector in profile view is not as reliable as frontal view, because the classifier relies on block features. It is interesting to note that the ear in frontal view is an outline feature, but in profile view it becomes a block feature. It would therefore make sense to use the ear as a key feature in profile view. Ear detection using cascaded Adaboost from a profile view is discussed in [12].

If the OPENCV face detector is trained with profile faces, the classifier has to attempt to learn the background variability beside the profile edge. The OPENCV Haar cascade provided for profile faces works reasonably well against a plain background, but to provide reliable detection against variable backgrounds, it would need to be trained with a much larger data set with highly random backgrounds [5].

Viola-Jones requires a very large set of training data. There would need to be thousands of positive examples of faces, and tens of thousands of non-faces. When training a cascade, there should be many more negative examples than positive examples, so that there are enough false positives detected by the early stages of the cascade that can be passed on to train the later stages of the cascade. The data should be well-separated (for example, not mixing training sets in different poses) and well-segmented (box boundaries should be consistent across the training set). These considerations mean that retraining the classifier is a formidable problem.

Note: Feature extraction from a profile image was demonstrated in principle using MATLAB code, but due to the issues discussed in this section, we did not develop a full implementation in OPENCV.

1.4 Back-End

The feature set extracted for each face is very small — simply an (x, y) coordinate for each reference point and feature key point. These points are labelled and sent to the back-end for analysis (inside a TCP/IP packet). Labels could include camera number, time-stamp, frame number, face bounding box within frame and an object tracking ID (if face tracking using particle filtering has been implemented).

In the back-end, a feature vector is calculated from the set of feature points, and this feature vector is compared to the database of enrolled people using a SVM (Support Vector Machine).

1.4.1 Feature Vector

To ensure robustness against changes in scale and rotation, only ratios of lengths and angles are stored in the feature vector. First, the distances between the reference points and the other key points are calculated, and expressed as a ratio to the reference length. As all distances are expressed as ratios, they are invariant to changes in scale.

Second, the angles between the reference points and other key points are calculated. The orientation of the baseline is used as the horizontal axis for the coordinate system of the feature vector. *i.e.*, the coordinates of the key points must undergo a matrix transformation into the new coordinate system. Normalising the key points to the baseline means that the feature vector is invariant to rotations in-plane.

The calculated angles do not themselves form part of the feature vector. Rather, the feature vector stores the difference between tangents of the angles formed at each reference point. Taking the difference between tangents means that the feature vector is invariant to rotations out-of-plane. A proof of this result can be found in [33].

The feature vector for the profile view is calculated in the same manner as the frontal view. The key point coordinates are transformed into a coordinate system using the profile baseline as the horizontal axis. The lengths and angles between the reference points and the other key points are calculated. These are encoded in the feature vector as ratios of the profile reference length and difference between tangents of angles. As with the frontal model, the feature vector is robust to changes in scale and pose (rotation).

1.4.2 Feature Vector and Matching

The set of reference points and feature key points is transmitted to the back-end over TCP/IP. The back-end calculates the feature vector from this set of points following this algorithm:

1. Calculate the gradient of the baseline (the line connecting the irises)
2. Use the gradient to calculate a rotation matrix to normalise the coordinate space, so that the baseline is parallel to the horizontal axis. This ensures that the feature vector is invariant to rotations in-plane
3. Rotate all the points about the left iris
4. Find the face centre line by taking the midpoint between the nostrils
5. Find the intersection of the face centre line and the baseline
6. Normalise the coordinate space so that the intersection of the baseline and face centre line is $(0,0)$. A graphical representation of the normalised feature points is shown in figure 1.19.
7. Calculate a set of lengths connecting the intersection point and the irises to the key feature points. These lengths are expressed as ratios of the baseline. This ensures invariance to scale
8. Calculate a set of tangents of angles between the intersection point and the key feature points. The feature vector stores only differences between these tangents. This ensures invariance to rotations out-of-plane[11]

[11] see [33] for proof

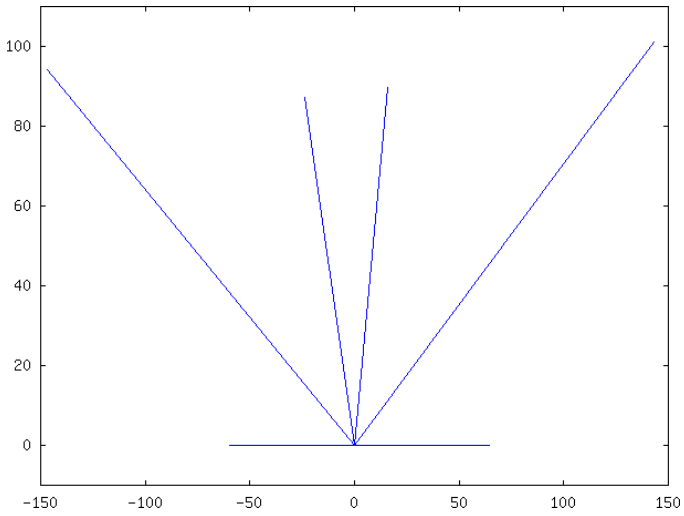

Fig. 1.19 Normalised feature points for four feature key points (two nostrils and two ear tips). The baseline is the horizontal line at the bottom and represents the line that joins the irises.

A set of feature vectors was calculated over a subset of the test videos and used to train the OPENCV SVM. The results of training were saved as an XML file that was used for verification and testing (see section 1.5.4).

1.5 Results

The aim of our experiments was to detect faces in a video stream, and extract a reliable set of features as rapidly as possible. The extracted features were then tested for discrimination between different subjects using a Support Vector Machine (SVM).

The process of face detection and local feature extraction is summarised in figure 1.3. It can be seen that a candidate face image can be rejected by the system at a number of points. In order to be passed to the back-end, the face candidate must pass skin detection, the nested Haar cascades for global and local features and a suitable set of local features must be identified and extracted. As our highest considerations were speed and reliability, we simply dropped any frames that could not pass all these tests. During the experiments, we attempted to reduce the number of dropped frames without compromising reliability.

Our feature extraction algorithms were developed and tested using still images from the FERET database. In order to test how our programs worked on video streams, we used a database of videos of speakers in different poses from CSIT (see section 1.1.3).

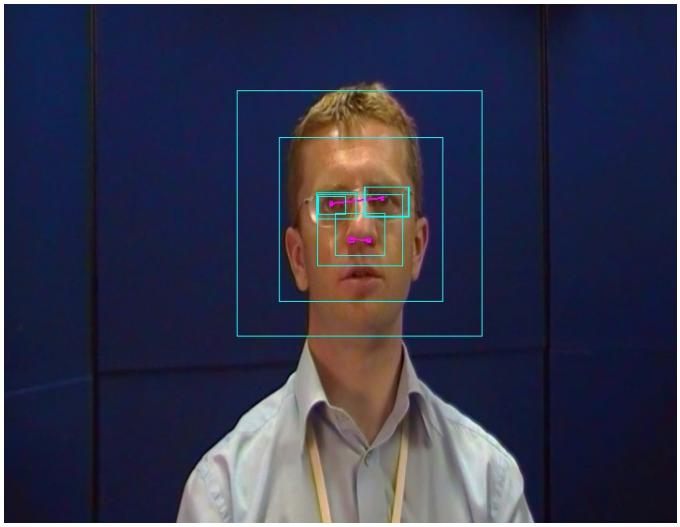

Fig. 1.20 Face matched at multiple scales using OPENCV Haar Cascade face detector

1.5.1 Skin Detection

The OPENCV adaptive skin detector was used to filter the image and to pass face candidate regions (rather than the whole image) to the Haar cascade detector. Our results show that performing skin detection on the video frames before passing them to the Haar cascade improved both speed and reliability.

Use of skin detection helped to eliminated false positives (non-face areas that are detected as faces). This could also include faces which are not real human faces (*e.g.*, a black-and-white photo of a face on a poster or T-shirt).

Sometimes the OPENCV Haar cascade detector finds the same face object at different scales (see figure 1.20). By searching skin regions, we can narrow the search to find the most meaningful match.

We experimented with using windows of different sizes around detected skin patches. The results are shown in figures 1.21 and 1.22. It can be seen that constraining the search area too tightly reduced the accuracy of the results. This is because areas of shadow to the sides of the face are sometimes not detected as skin. The Haar cascade detector therefore needs an area wider than the skin patch for a successful detection. As we increased the search window around the skin patches to include these shadowed areas, the accuracy of the Haar detection gradually increased until (at around 30–50 pixels) it was as accurate as searching on the whole image.

Obviously the greatest speedup is achieved with a smaller search window. As we increased the size of the search window, the speedup decreased exponentially. However, a window of 30–50 pixels is still measurably faster than not using skin detection.

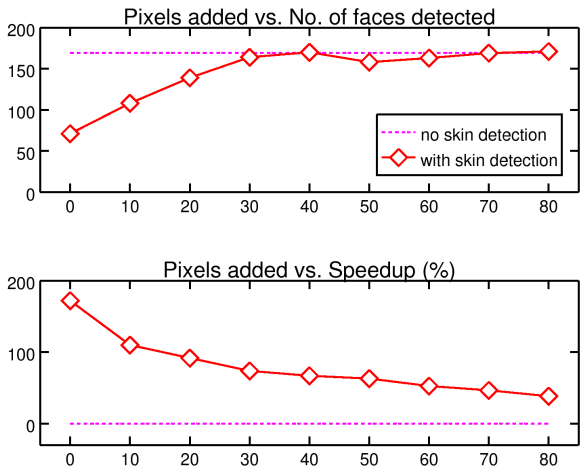

Fig. 1.21 Affect of Skin Detection on Performance: Darryl

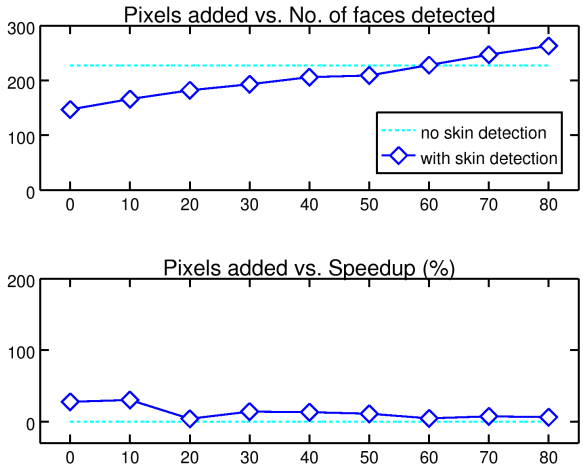

Fig. 1.22 Affect of Skin Detection on Performance: Adrian

Skin detection is also vulnerable to false positives and this reduces performance. This is illustrated in figure 1.22, as the colour of Adrian's clothing is within the range of detected skin tones (see figure 1.27; the outer bounding box indicates the detected skin region that is passed to the Haar cascade for face detection). In this case, there is no performance improvement by using skin region detection.

The strategy of skin detection would not be of any benefit if large areas of the background were skin-coloured. In a real system, whether or not to use skin detection could be configured at each sensor.

For our experimental system, we chose a window of 40 pixels around skin regions as giving the best compromise between performance and accuracy.

1.5.2 Face Detection

The most important features for the OPENCV face detector are the eyes and nose. During informal experiments, it was observed that occluding the eyes or nose prevents a face from being detected. The mouth is a less important feature. It is often possible to occlude the mouth without affecting face detection. In any case our system does not extract features from the mouth area as we restricted our system to detecting features in the non-deformable part of the face.

As illustrated in figure 1.3, if we did not detect a suitable face and local features within a frame, that frame would be discarded. Figures 1.23 and 1.24 show how many face candidates are rejected at each stage of the detection process.

Note that the OPENCV face detector returns many false positives. In our tests, each frame of video contained exactly one face, but figure 1.23 shows that the number of faces returned by the Haar cascade is greater than the number of frames. The results show that skin detection slightly reduced the number of false positives (while speeding up the overall detection rate).

More face candidates are rejected by the nested Haar cascades which detect local features (eyes and nose). If the system cannot detect two eyes and a nose, the face candidate is discarded. Finally, we identify reference points and feature key points within the local features. If these cannot be accurately identified, the face candidate is discarded. Recognition is attempted only on face candidates which pass all stages of the detection process. Thus, we are discarding around 60%–70% of face candidates in order to improve recognition accuracy.

1.5.3 Feature Extraction

1.5.3.1 Morphological Operations

In order to optimise blob detection, we experimented using a number of convolution kernels:

- Custom kernels for erosion and dilation
- Built-in convolution kernels for OPENCV
- Morphological transformations for opening and closing the image

The morphological operations of *closing* and *opening* an image predefined in OPENCV represent a combination of erosion and dilation. The effects are as follows:

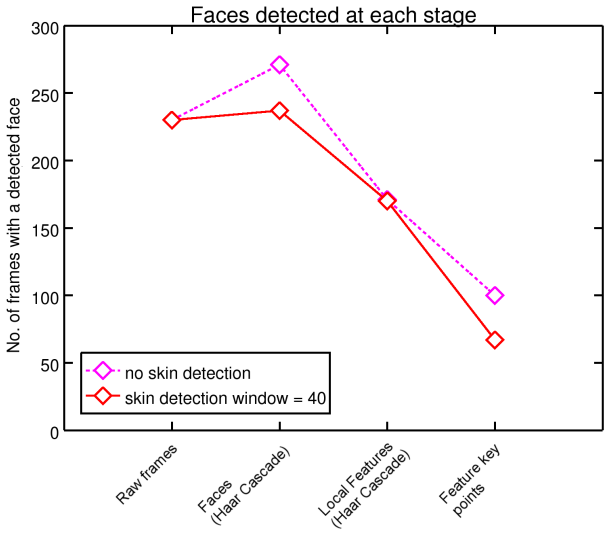

Fig. 1.23 Rejection of unsuitable face candidates at each stage of the detection process: Darryl

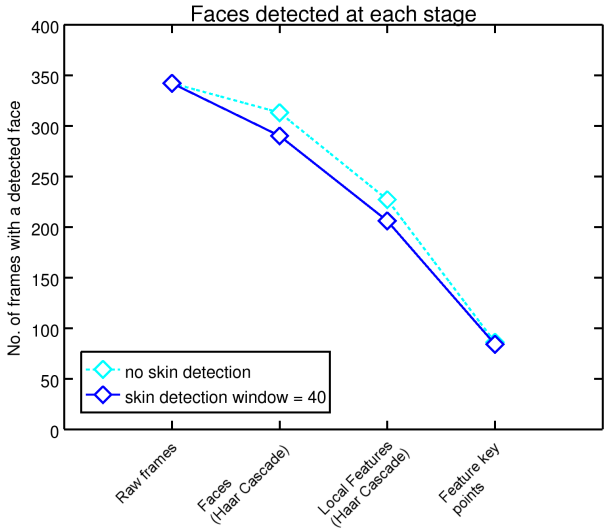

Fig. 1.24 Rejection of unsuitable face candidates at each stage of the detection process: Adrian

Opening an image results in removing small bright regions of the image. The larger bright regions are isolated but retain their size. We tried opening the image because a relatively small bright region (appearing as an artefact of wearing a nose ring) was interfering with our algorithm of nostril detection.

Closing an image results in joining bright regions within the image. The dark regions remain dark and their size is unchanged. We tried this, aiming to provide a better contrast between the nostril/pupils and the brighter areas that surround them.

Opening then closing an image should theoretically discard the smaller brighter regions and then join all the larger brighter regions while keeping the dark regions almost unchanged. Although this makes the nostrils more obvious to the human eye, OPENCV blob detection (based on contour detection) fails because this operation has the effect of destroying edges.

The test video images were convolved with 12 different kernels, and the performance of each was compared. Figure 1.25 illustrates how the choice of convolution kernel affects the accuracy of feature detection. Each kernel was also compared for its effect on computation time, as shown in figure 1.26[12].

Our experiments showed that convolving the images with a rectangular kernel of size 2×2 provided the most accurate detection of key features (pupils and nostrils). There was a slightly higher cost in terms of processing time, but this was deemed to be an acceptable trade-off.

1.5.3.2 Filtering Local Features

Local features were detected using a nested Haar cascade. Often the Haar cascade would return one or more false positives for each local feature as well as the correct location of the feature. It was therefore necessary to filter the local features to determine which candidate was the correct one.

Several general rules were set when filtering the features:

- The right eye should be in upper right part of the face
- The left eye should be in the upper left part of the face
- The nose should be in the central region of the face. Sometimes the bounding boxes for the nose and the bounding boxes for the eyes can briefly overlap.

The proportions of the human face have been studied for many years, in particular by surgeons specialised in reconstructive and plastic surgery. Thus, medicine has determined that the height of the nose relative to the frontal face should be around 47% [23]. The frontal face Haar cascade often crops the lower part of the face (see figures 1.27 and 1.28). This cropping is very unpredictable, so the face height is unreliable for filtering features. However, the nose is never cropped. Since the nose

[12] The time in seconds refers to the time to process the entire video sequence. The video sequences were 342 and 230 frames respectively, so our system is somewhat slower than real time. Real time speeds could easily be achieved in a production system by exploiting hardware acceleration or parallel processing.

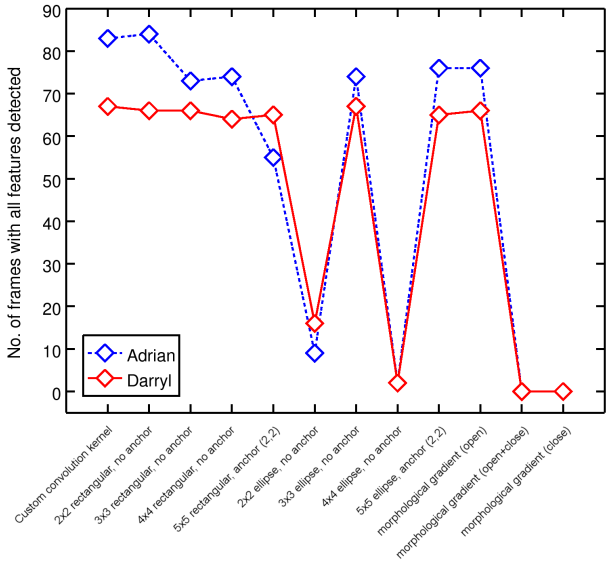

Fig. 1.25 Analysis of choice of erosion/dilation structures on detection accuracy

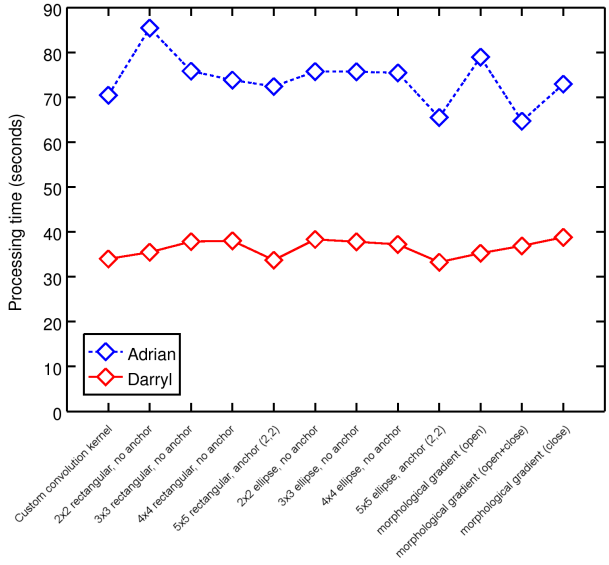

Fig. 1.26 Analysis of choice of erosion/dilation structures on performance

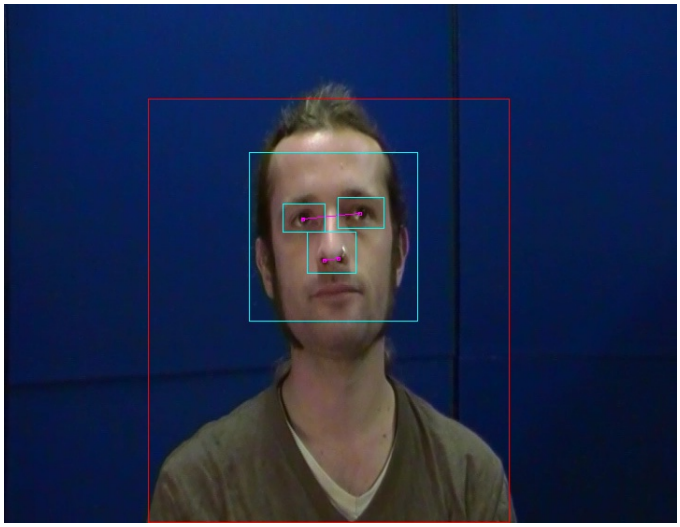

Fig. 1.27 Face detected using Haar Cascade, but upper and lower regions are cropped

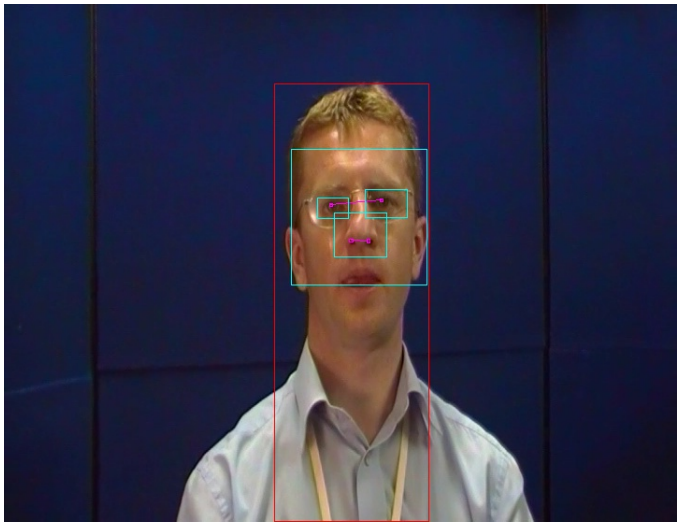

Fig. 1.28 Face detected using Haar Cascade, but upper and lower regions and ears are cropped

height is not affected by cropping, we tried to use it in defining a ratio that would give the approximate position of the top part of the eyes.

The human frontal face can be divided into three horizontal regions. The aesthetic ideal is that the three regions are the same height, but in our system we allow some flexibility. Since we are interested in determining a relation that would give us the position of the eyes relative to the top of the face, we are only interested in the upper region. This lies between the trichion[13] and glabella[14]. This is approximately 33% of the entire facial height but since the Haar detection of the frontal face crops this part, we calculate this region as being between 20–30% of the face height. The correct position of the eyes should be detected just under the glabella. So the position of the eyes on y axis should be approximately 0.2–0.3 $\times face\ height$ (1) and the height of the nose should be approximately $0.47 \times face\ height$ (2). By substituting $face\ height$ from (2) into (1) it follows that the eyes are expected at approximately 0.42–0.63 $\times nose\ height$.

We conducted experiments to determine the coefficient that works best within this interval. Table 1.1 shows the number of faces (containing two eyes and a nose) found in the video stream for different values of the coefficient.

The value 0.53 was chosen for the facial proportion coefficient, because it provided good detection rates and no false positives. 0.5 provides even better detection rates but was considered too close to the value that outputs false positives.

In our test system, the position of the top of the nose bounding box was approximated using a fixed number of pixels. The detection was calibrated for the size of the faces in the test videos. In a real system, this parameter would have to be automatically calibrated to work on a wider range of images.

The proportions that were used to choose the best nose and best eyes from all candidates are shown in figure 1.29.

1.5.4 Feature Vector and Recognition System

Face recognition in the back-end was implemented using the OPENCV SVM class. A set of training videos was processed by the front-end. Feature points from the training set were sent to the back-end, which generated feature vectors and dumped them into a file. A matrix was initialised with each feature vector as a row and we specified how many rows represented training data for each subject. The result of training was saved in an XML database.

The SVM was trained for two subjects using one video for each subject. In the training videos, the subjects were looking straight ahead. The video for Adrian was 13 seconds long and 67 faces were extracted and passed to the SVM for training. The video for Darryl was 9 seconds long, and 83 faces were extracted. The number of frames extracted for Adrian is lower because in many frames his eyes are closed,

[13] the point where the hairline meets the midpoint of the forehead.

[14] reference point in anthropology representing a smooth elevation in the frontal bone just above the bridge of the nose.

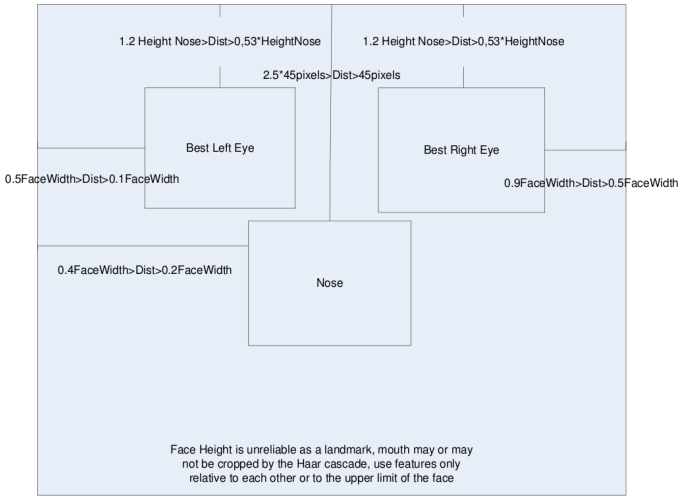

Fig. 1.29 Feature Extraction: Selection of best local features from within the face bounding box

Subject	Coefficient	No. of faces detected
Adrian	0.48	316[15]
	0.5	280
	0.53	205
	0.55	128
	0.6	21
Darryl	0.48	170
	0.5	170
	0.53	170
	0.55	168
	0.6	117

Table 1.1 Effect of coefficient for facial proportions on detection accuracy. The Adrian video had 342 frames and the Darryl video had 240 frames.

so those frames are discarded. Also, Adrian's nose ring made accurate feature extraction more challenging for the front-end!

The feature vector in our experimental system is of low dimensionality, with only six features. Figure 1.30 illustrates the discriminative capacity of our system, showing a comparison of features 3 and 6 for the two subjects. Features 1, 2, 4 and 5 measured ratios of distances between features (relative to the reference length), whereas features 3 and 6 measured the differences of tangents between the reference points and feature key points. Features 3 and 6 were therefore the most robust to

[15] 316 detections includes some false positives — in some cases, eyebrows are detected as eyes

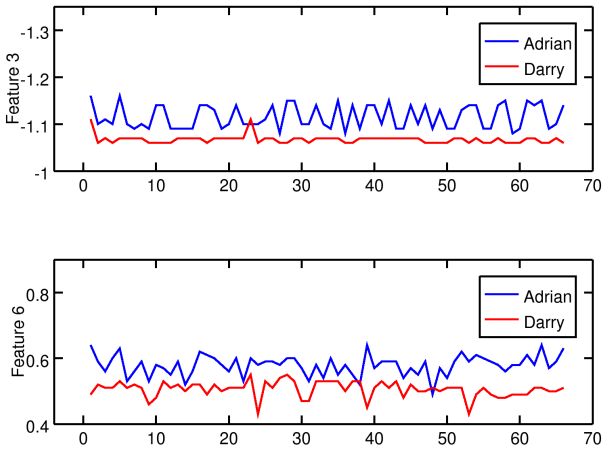

Fig. 1.30 Discriminative capacity of the feature vector: comparison of two features for two subjects

Subject	Faces found (front end)	Faces identified (back end)	False positive	False negative
Adrian	20	20	0	0
Darryl	38	38	0	0

Table 1.2 Results of identification test for two subjects

rotations out-of-plane and provided the greatest discrimination between the subjects in our experiments.

We demonstrated that it is possible to calculate more lengths and tangents between the feature points we have already extracted. A production system would add additional feature points to provide a feature vector of larger dimensionality and therefore capable of greater discrimination.

Testing was conducted using a different video from the training set for each subject. The video for Darryl was 9 seconds long and he was rotated 10% from frontal view. The video for Adrian was 11 seconds long and was facing straight ahead. The videos were processed by the front-end and feature points were sent to the back-end for identification. The back-end calculated the feature vector for each frame and returned identification results as shown in table 1.2.

These tests can be considered as a proof-of-concept. It would require a larger set of test data to do an accurate comparison of the discriminative capacity of the feature vector.

It was not possible to test on rotations larger than 10% as we were using the ear tips as a feature, and at greater than 10% rotation, one ear is occluded. This could

be compensated for by adding more features and by using missing feature theory to ignore the effects of occluded features.

1.6 Conclusions

This project has demonstrated a system for face recognition from surveillance video. The system was demonstrated to be robust to changes in illumination, scale, facial expression and reasonably robust to occlusions and changes in pose. Attention was given to performance considerations, and the system can operate in real time. All of these features make this approach suitable for a video surveillance application.

One of the notable findings was the improvement in both performance and accuracy of the OPENCV face detector, when the Viola-Jones face detection was combined with an analysis of colour skin-tone information. Colour information could be exploited further to improve the detection of local features as discussed in [10]. Our feature-detection algorithms were demonstrated to be invariant to changes in lighting direction.

The local features that we used were irises, nostrils and ear-tips. Irises and nostrils are reasonably straightforward to extract in frontal view, although reflected light from glasses or a nose-ring can degrade performance. Ears are more difficult to locate as Viola-Jones' method is better at detecting "blocky" features rather than outline features. Ears can also be occluded by hair, earrings or self-occluded by a rotation of the face. Nonetheless, it was observed that the difference in tangent angles to the ear-tips was one of the most discriminant features in the feature vector.

In profile view, the key features were the location of the iris and the shape of the nose profile. The iris may not be available if the person is wearing glasses, as the stem of the glasses usually occludes the eye. It is easy to detect the ear in profile view (if it is not occluded) as it becomes a "blocky" feature when viewed from the side.

In a real-world system, it would be necessary to select more features than we have outlined here to provide sufficient discrimination between people enrolled on the system. Suitable features could include corners and edges on facial features such as the eyebrows and mouth. We have already mentioned that the eyebrows are one of the most important features in human facial recognition [29]. Features in the non-deformable part of the face should be accorded more weight, to provide robustness to changes of expression. We have also mentioned that the full set of features may not be available in every case. It may be necessary to select a different set of features for different subjects. Missing-feature theory could be applied to this problem [17].

It is unknown how many features are required to provide discrimination over a larger population. This could be the subject of a future study. We envisage that this type of system would be used in a controlled environment (such as a Secure Corridor in an airport) where the population is of limited size and people can be enrolled onto the system as they enter the Corridor.

It is likely that the best results will be achieved by combining this approach with other methods. Colour information was used in detection but not as part of the recognition system. [29] states that pigmentation is an important recognition cue for humans (at least as important as shape), so it would be natural to include skin pigmentation, hair colour or eye colour as part of the feature vector.

Our shape-based approach could also be combined with appearance-based approaches in a hybrid detection system. Just like Bertillon in the 19th century, local feature recognition could be used to narrow the search space and then global features could be used to get an exact match. It is also possible to use appearance-based methods on local features (Eigeneyes, *etc.*). In this case, our method could be used to perform pose estimation. [19] discusses how to improve face recognition using a combination of global and local features.

The accuracy of facial recognition can also be improved by combining face detection with other forms of detection using multi-modal fusion. Gait analysis would be a likely candidate as it does not require any additional sensors.

Finally, it is worth mentioning that this approach to machine vision also has non-security applications, for example the recognition of human faces by a robot or computer game.

Appendix: Further Implementation Details of the Experimental System

The system used to conduct the experiments in this chapter was written in C++ using the OpenCV libraries. The source code is available for free download from our website (http://www.electriceye.org.uk), under the terms of the GNU General Public License. The website also has some short videos illustrating our experimental results.

An overview of the system architecture was described in section 1.2 and the relationship between front-end camera nodes and the back-end database and classifier was shown in figure 1.2. Figure 1.31 shows a more detailed view of a single node and its communication with the back-end. The node detects faces and extracts a feature vector for each face. Object tracking could be carried out by the nodes, so that

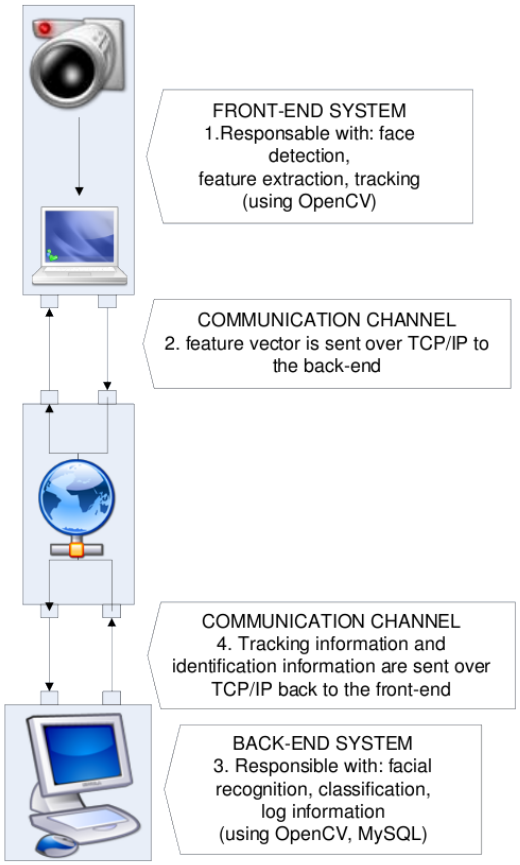

Fig. 1.31 Distributed System Architecture: Communication between Front-End and Back-End

multiple face samples can be tagged with an identifier, thus attributing the samples to the same person. (This feature was not implemented in our system). The feature vectors are then packaged for transmission to the back-end using TCP/IP.

In section 1.3.3, we described how the front-end should attempt to match faces to both a frontal and a profile model of the individual. In the design of our system, each match would be performed by an independent thread, as shown in figure 1.32. The "second thread" in the front-end is in fact two separate threads, one detecting left-facing profiles and one detecting right-facing profiles. Thus, three threads would search for frontal faces, right profiles and left profiles, respectively. A fourth thread handles all the communication with the back-end. (Profile detection was not implemented in our system; see section 1.3.3.5 for discussion).

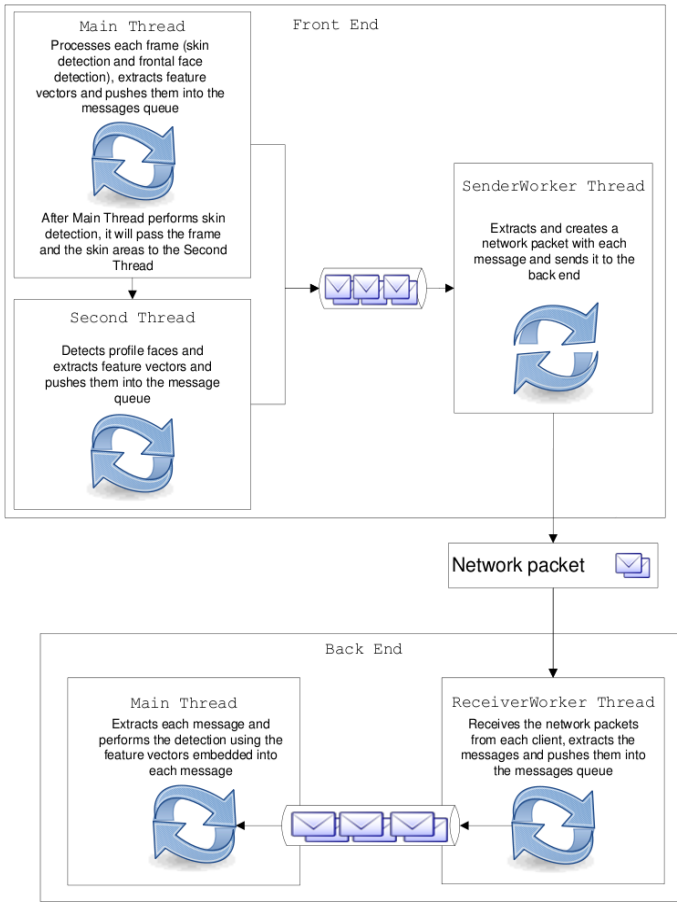

Fig. 1.32 Front End: Multi-threaded approach to feature extraction and matching against frontal face model and profile face model

References

1. Alex, M., Vasilescu, O., Terzopoulos, D.: Multilinear analysis of image ensembles: TensorFaces. In: COMPUTER VISON — ECCV 2002, PT 1, *LECTURE NOTES IN COMPUTER SCIENCE*, vol. 2350, pp. 447–460 (2002)
2. Bartlett, M., Movellan, J., Sejnowski, T.: Face recognition by Independent Component Analysis. IEEE TRANSACTIONS ON NEURAL NETWORKS **13**(6), 1450–1464 (2002). DOI 10.1109/TNN.2002.804287
3. Belhumeur, P., Hespanha, J., Kriegman, D.: Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE **19**(7), 711–720 (1997)
4. Bentham, J.: Panopticon; or, the inspection-house: Containing the idea of a new principle of construction applicable to any sort of establishment, in which persons of any description are to be kept under inspection; and in particular to penitentiary-houses. http://oll.libertyfund.org/ (1843)
5. Bradski, G.R., Kaehler, A.: Learning OpenCV (2008)
6. Daugman, J.: High confidence visual recognition of persons by a test of statistical independence. IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE **15**(11), 1148–1161 (1993)
7. Forsyth, D., Ponce, J.: Computer vision: a modern approach. Prentice Hall, Upper Saddle River, N.J. ; London (2003)
8. Freund, Y., Schapire, R.: A decision-theoretic generalization of on-line learning and an application to boosting. JOURNAL OF COMPUTER AND SYSTEM SCIENCES **55**(1), 119–139 (1997)
9. Funahashi, T., Fujiwara, T., Koshimizu, H.: Hierarchical tracking of face, facial parts and their contours with PTZ camera. In: 2004 IEEE International Conference on Industrial Technology (ICIT), pp. 198–203. IEEE (2004)
10. Hsu, R., Abdel-Mottaleb, M., Jain, A.: Face detection in color images. IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE **24**(5), 696–706 (2002)
11. Isard, M., Blake, A.: Condensation — conditional density propagation for visual tracking. INTERNATIONAL JOURNAL OF COMPUTER VISION **29**(1), 5–28 (1998)
12. Islam, S., Bennamoun, M., Davies, R.: Fast and fully automatic ear detection using cascaded AdaBoost. In: 2008 IEEE WORKSHOP ON APPLICATIONS OF COMPUTER VISION, IEEE Workshop on Applications of Computer Vision, pp. 205–210. IEEE (2008)
13. Jiang, R.M., Crookes, D.: Multimodal biometric human recognition for perceptual human-computer interaction (draft). IEEE Transactions on Systems, Man and Cybernetics (2010)
14. Kawaguchi, T., Rizon, M., Hidaka, D.: Detection of eyes from human faces by hough transform and separability filter. ELECTRONICS AND COMMUNICATIONS IN JAPAN PART II-ELECTRONICS **88**(5), 29–39 (2005). DOI 10.1002/ecjb.20178
15. Klauser, F.: Interacting forms of expertise in security governance: the example of CCTV surveillance at Geneva International Airport. British Journal of Sociology **60**(2), 279–297 (2009). URL 10.1111/j.1468-4446.2009.01231.x
16. Lienhart, R., Maydt, J.: An extended set of haar-like features for rapid object detection. In: 2002 INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, VOL I, PROCEEDINGS, IEEE International Conference on Image Processing (ICIP), pp. 900–903. IEEE Signal Proc Soc, IEEE (2002)
17. Lin, J., Ming, J., Crookes, D.: A probabilistic union approach to robust face recognition with partial distortion and occlusion. In: 2008 IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING, VOLS 1-12, International Conference on Acoustics Speech and Signal Processing (ICASSP), pp. 993–996. IEEE (2008)
18. News, B.: 1,000 cameras 'solve one crime'. http://news.bbc.co.uk/1/hi/8219022.stm (24 August 2009)

19. Nor'aini, A., Raveendran, P.: Improving face recognition using combination of global and local features. In: 2009 6TH INTERNATIONAL SYMPOSIUM ON MECHATRONICS AND ITS APPLICATIONS (ISMA), pp. 433–438. IEEE (2009)
20. Norris, C.: The Maximum Surveillance Society: the Rise of CCTV. Berg, Oxford (1999)
21. Norris, C., Armstrong, G.: Space invaders: The reality of a CCTV control room in Northern England raises the old question, "Who guards the guards?". INDEX ON CENSORSHIP **29**(3), 50–52 (2000)
22. P, A.: Surveillance at the airport: surveilling mobility/mobilising surveillance. ENVIRONMENT AND PLANNING A **36**(8), 1365–1380 (2004). URL 10.1068/a36159
23. Papel, I., Frodel, J.: Facial plastic and reconstructive surgery. Thieme, New York (2002)
24. Pass, A., Zhang, J., Stewart, D.: An investigation into features for multi-view lipreading. IEEE International Conference on Image Processing (ICIP). IEEE (2010)
25. Register, T.: IT contractors convicted of uk casino hack scam. http://www.theregister.co.uk/2010/03/15/uk_casino_hack_scam/ (15 March 2010)
26. Rosen, J.: A cautionary tale for a new age of surveillance. http://www.nytimes.com/2001/10/07/magazine/07SURVEILLANCE.html (7 October 2001)
27. Shapiro, L.G.: Computer vision. Prentice Hall (2001)
28. Sharkas, M., Abou Elenien, M.: Eigenfaces vs. Fisherfaces vs. ICA for face recognition; a comparative study. In: ICSP: 2008 9TH INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING, VOLS 1-5, PROCEEDINGS, pp. 914–919. IEEE (2008)
29. Sinha, P., Balas, B., Ostrovsky, Y., Russell, R.: Face recognition by humans: Nineteen results all computer vision researchers should know about. PROCEEDINGS OF THE IEEE **94**(11), 1948–1962 (2006). DOI 10.1109/JPROC.2006.884093
30. Turk, M., Pentland, A.: Face recognition using Eigenfaces. In: 1991 IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, pp. 586–591. IEEE (1991)
31. Viola, P., Jones, M.: Robust real-time face detection. INTERNATIONAL JOURNAL OF COMPUTER VISION **57**(2), 137–154 (2004)
32. Wiskott, L., Fellous, J., Kruger, N., vonderMalsburg, C.: Face recognition by elastic bunch graph matching. IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE **19**(7), 775–779 (1997)
33. Xu, Z., Wu, H.R.: Shape feature based extraction for face recognition. In: ICIEA: 2009 4TH IEEE CONFERENCE ON INDUSTRIAL ELECTRONICS AND APPLICATIONS, VOLS 1-6, pp. 3034–3039. IEEE (2009)
34. Zhao, W., Chellappa, R.: Face processing (2006). URL http://www.loc.gov/catdir/enhancements/fy0645/2006296212-d.html
35. Zhao, W., Chellappa, R., Phillips, P., Rosenfeld, A.: Face recognition: A literature survey. ACM COMPUTING SURVEYS **35**(4), 399–459 (2003)
36. Zhou, H., Yuan, Y., Sadka, A.: Application of semantic features in face recognition. PATTERN RECOGNITION **41**(10), 3251–3256 (2008). DOI 10.1016/j.patcog.2008.04.008