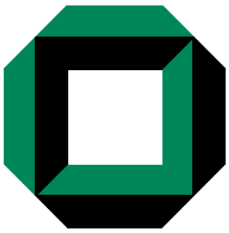


University of Karlsruhe (TH)
Interactive System Labs
Prof. Dr. Alexander Waibel

Studienarbeit

Informatik

Thema: Open-set Face Recognition

Lorant Szasz-Toth - loranttoth@gmail.com

JANUAR 2008

Betreuer: Prof. Dr. Alexander Waibel
Dr.-Ing. Rainer Stiefelhagen
Hazım K. Ekenel

Abstract

This work presents a local appearance-based approach to open-set face recognition. The open-set recognition task is formulated as a multi-verification problem. The recognition task is carried out as a series of verifications, every new probes' identity is established by performing identity verifications against each known subject in the gallery. The classification is based on local appearance-based face recognition approach using DCT. Input images are divided into local blocks to which the discrete cosine transform is applied. A feature vector is then generated by combining selected local feature coefficients and classification is performed using either support vector machines or nearest neighbor method. Progressive accumulation of confidence scores from each verifier is used to enhance video-based identification. Additionally, a data set has been collected in front of the entry to an office to evaluate the performance of the proposed approach.

Contents

List of Figures

List of Tables

1 Introduction

Face recognition plays an important role in human-machine interaction. Especially smart environments need robust and unobtrusive means of identifying interaction partners in order to achieve the goal of assisting the user without interference.

Face recognition is a natural and innate ability of humans and is used daily by everybody. Usually, we visually identify our communication partner. This further influences our behavior and helps to improve interaction. Once the identity has been established, it is possible to tap background knowledge in order to better interpret intentions, statements and gestures or modify the own behavior.

Allowing a system to identify its interaction partners helps it to adapt to the user. A sample application domain is customized services based on the user's preferences. For example, a car can recognize its driver and adjust cockpit and seat settings to the user's taste or a smart living room that adjusts ambient lighting and favorite TV channels.

This work focuses on an open-set face recognition scenario. Open-set recognition is the task of identifying a set of known people from an open set of possible probes, closed-set recognition on the other hand assumes that every probe is from a closed set of subjects that the system was trained on. The difference is therefore that arbitrary people may be presented to the open-set system which has to decide whether the person is known or unknown and then, in case the probe is known, determine the identity. Generally, most identification problems are not of a closed-set nature, therefore open-set recognition presents an approach that is more applicable to real-world problems. Nonetheless, open-set recognition is not a common research topic.

The first step to building a biometric system for face recognition is to extract biometric information from samples of the subject that the system later has to recognize and to store these feature templates in a database (or gallery) - these belong to the known set. The set of unknowns are all those probes presented to the system that do not have a mate in the gallery. Then, during matching, the system is presented a probe sample that it has to classify.

The main challenges lie in changing illumination, varying pose and varying appearance. Change of illumination is a problem because different lighting levels may yield very different gray value distributions in faces, i.e. the same face looks very different under artificial lighting and in direct sunlight. So much that two different faces may look more similar under the same lighting than the same face under different lighting conditions.

Pose variation occurs very frequently because the user cannot be expected to look straight into the camera at all time without moving, tilting or turning the head. The head may rotate out of image plane, if the user looks at another person instead of the camera, or in plane, if the user tilts his head. All these distortions shift the positions of local features because for example in a slightly out of plane rotated face the eyes lie closer.

Appearance may also change over time. For one, people may be wearing different accessories like glasses or sunglasses, may grow a beard or makeup. All these change the overall appearance of the face and they have to be taken into account. Additionally,

parts of the face may be covered, e.g. the mouth by the user's hand. And different facial expressions like laughing or speaking may also alter the appearance. A number of approaches have been developed to recognize and identify faces. These range from facial feature based methods, to appearance based models. Some of these systems perform quite well in closed set identification tasks but open set recognition is more challenging.

Face recognition or identification is the task of establishing a person's identity by means of a presented face image. Face detection is the task of determining whether a given image contains faces and finding the locations and sizes of all faces. Therefore face detection is the first step in order to find and extract the face, followed by registration where the detected face is extracted, cropped and aligned. Then, usually a feature extraction step that reduces the data dimensionality follows because an $N \times M$ sized image spans a classification space of $N \times M$ dimensions, and dimensionality increases the need for more training data. Consequently, the feature extraction usually maps the input image into a lower-dimensional space. After feature extraction classification takes place. This is the final step in the recognition process where the extracted feature is compared with the previously learned or stored features in a database.

Different classes of face recognition problems exist.

1. Closed-set recognition: For a fixed, finite set of known individuals decide who the presented sample is most similar to.

2. Face verification: Given a sample and a claimed identity decide whether the claim is correct.

3. Open-set recognition: For an arbitrary set of individuals (known and unknown) decide whether the given sample is known and, if known, which person he/she is.

Closed-set face recognition works on a fixed set of known subjects and has to distinguish just between these known individuals. Open-set face recognition is a more challenging problem than closed-set recognition because an additional decision has to be made whether the person is known or unknown. Open-set recognition can be seen as an extension of face verification to n subjects where every subject has to be verified against an open set of impostors. This work implements open-set face recognition as a multi-verification task. This is done by building a face verification classifier for every known subject. Then, every new probe is tested against these verifiers to establish the identity.

1.1 Previous work

According to [11] face recognition algorithms can be classified as feature-based and appearance-based algorithms. Feature-based algorithms extract facial characteristics like proportions and distances of facial features. Appearance-based algorithms directly work with pixel intensities, e.g. neural networks as in [12]. Additionally, appearance-based algorithms can be divided into holistic and local approaches.

Holistic approaches represent the entire face in a single feature vector whereas local appearance based approaches work on smaller portions of the face - either dividing the

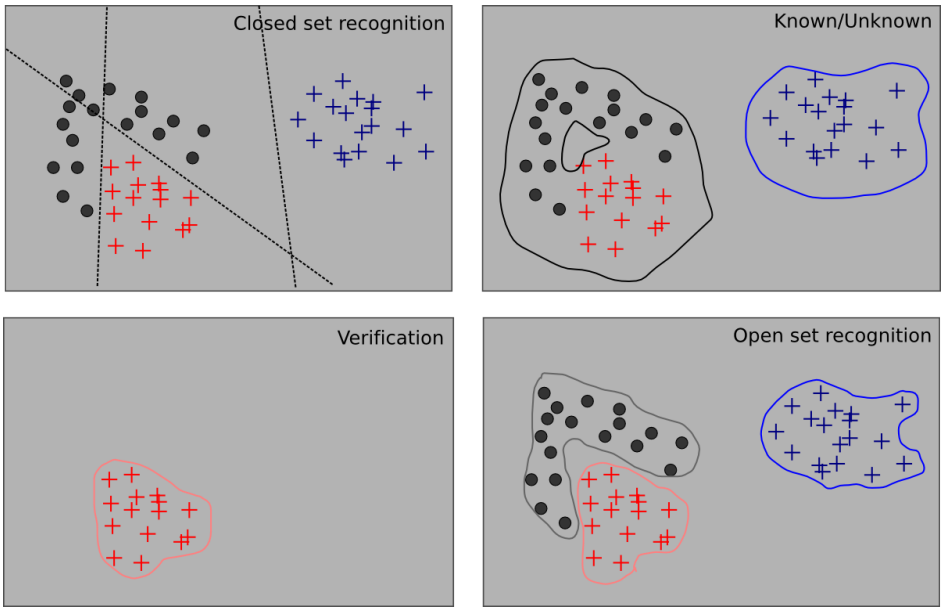


Figure 1: Face recognition tasks as described in [8]

whole face into regions which are analyzed independently or focusing on prominent regions such as eyes or nose.

1.1.1 Holistic approaches

Holistic face recognition approaches analyze the entire face at once. The most well-known holistic approach is the eigenfaces approach [9]. A Karhunen-Loeve transform (KLT) - also known as principal component analysis (PCA) - is applied to a training set of previously aligned and cropped faces. Then, the first m principal components (eigenfaces) are selected thereby reducing the dimensionality of the recognition task. Now, new faces can be expressed as a linear combination of this eigenface base so comparing an unknown face with previously learned faces is just a matter of comparing coefficients that associate with each eigenface. Drawbacks of this approach are its sensitivity to lighting and pose variations as well as registration errors. Fisherfaces [6], [7] is a similar holistic approach that uses Fisher linear discriminant analysis (LDA) for dimensionality reduction. The eigenfaces approach maximizes the overall scatter, whereas fisherfaces uses class information and maximizes the ratio of between-class scatter to within-class-scatter. The eigenfaces approach aims at best reconstruction of face images in a mean square error sense. Fisherfaes on the other hand deals directly with classification.

1.1.2 Local appearance-based approaches

It has been found that holistic approaches are more sensitive to lighting and pose changes and suffer from occlusion and changes in facial expression. Therefore local appearance based approaches were introduced. Modular eigenfaces [9] for example that focuses on eye and nose regions only for recognition has been shown to outperform the holistic eigenface approach. Another example that is using local appearance-based features is [10]. There support vector machines are used as classifiers on local face components which partially overlap. The local component-based approaches outperformed holistic approaches using support vector machines as well.

1.1.3 Face verification

Face verification plays an important role in secure biometric applications and has been studied extensively. It is also important in this work because a multi-verification approach is used to implement open-set face recognition. Support vector machines were used to implement high-security verification systems, examples are approaches by Jonsson et al. [17], Kim et al. [19], Lee et al. [20], Phillips [22] and Cardinaux et al. [21].

Phillips [22] applies SVMs to face verification. The work introduces a parametrized decision surface in order to optimize the trade-off between the probability of correct verification P_V and false acceptance P_F, the decision surface $\{x \epsilon \mathbb{S} : wx+b = 0, (w,b)\epsilon \mathbb{S} \times \mathbb{R}\}$ is changed to $wx + b = \Delta$, where Δ allows variation of P_V and P_F. Classification is performed in difference space consisting of within-class differences modeling differences within the same class and between-class differences that model differences between different classes. Then, the SVM is trained to build a decision surface separating these classes. This SVM can be used to estimate similarities between two facial images.

Jonsson et al. [17] employ support vector machines and explore their discriminatory capabilities. Faces are both represented in principal component and linear discriminant subspaces for comparison. They show that SVMs can extract the relevant information and outperform benchmark approaches on non-discriminatory representation but lose this advantage if discriminatory features like those obtained by LDA are used. Unlike [22] this approach uses client-specific support vectors.

Kim et al. [19] build a verification framework considering real-world applicability. They try to limit memory consumption and consider easy removal and addition of clients. The approach is based on a feature set comprised of several PCA-based features and an additional edge map. The work describes the new features as eigenUpper, consisting of forehead, eyes and nose that tries to minimize influence of expression, and eigenTzone, consisting of eyes and nose that tries to compensate illumination. Instead of training one SVM per enrolled client an SVM monitor is trained to model an intra-/extra-person similarity space on eight different simple similarity measures. By using a similarity space and a single SVM memory consumption and computational overhead are reduced.

Cardinaux et al. [21] train SVM and MLP classifiers on features consisting of raw-pixel values of resized and normalized face images and a skin color distribution vectors. In

this case the MLP outperformed the SVM approach.

1.1.4 Open-set face recognition

Open Set Face Recognition Using Transduction is one of the few papers [4] that explicitly addresses the issue of open-set recongition.

2 Methodology

This section gives an overview of the basic principles and techniques used in this work.

2.1 Haar cascade classifiers

Haar cascade classifiers, developed by Viola and Jones (2001) [18], present a fast and efficient object detection framework. The framework utilizes Haar-like features and boosted classifier cascades. Haar-like features are simple rectangle features inspired by Haar basis functions. AdaBoost is then used to select the most promising Haar-like features efficiently. In order to further improve performance a cascaded structure was introduced. A cascade is a degenerate decision tree that assesses a limited combination of features at every stage. Early stages try to reject as many negative samples as possible while accepting nearly all positive samples. The advantage is then that only few search windows have to descend through the whole cascade.

2.1.1 Haar-like features

Haar-like features are simple rectangle features inspired by Haar basis functions. Computed features have advantages over using pixel intensities directly. Although far more rectangle features than pixels exist, once a set of discriminative features has been selected they can be evaluated faster and they reduce the classification complexity by utilizing a lower dimensional feature spaces and thus allow better results using a finite amount of training examples. Examples of these rectangle features can be seen below.

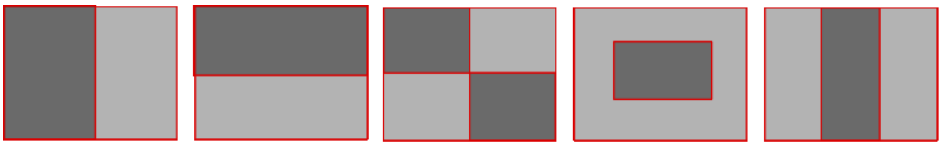


Figure 2: The following sample Haar-like features consist of up to four boxes whose gray value differences are computed

Haar-like features consist of up to four rectangular image regions and they are computed as differences of rectangular image regions. These features can be of arbitrary size and position within the search window. Therefore even relatively small search windows yield a large number of possible features.

Albeit being simpler than other possible feature representations Haar-like features have one distinct advantage. They can be computed very efficiently due to their simple nature. In order to keep computation overhead at a minimum so-called integral images were introduced.

2.1.2 Integral image

In order to compute rectangle features very rapidly the integral image is used. The integral image is a different representation of images that allows to compute intensity sums of rectangles very efficiently. Every point of the integral image corresponds to the sum of all intensities within the rectangle that is spanned from the image origin to the current point's location.

The integral image can be calculated according to

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{1}$$

where x, y are the integral image positions and $i(x', y')$ represents the image intensity value at (x', y'). This formula can be computed with a single pass over the image by temporarily storing the current row's intensity sum and referencing previously calculated rectangle sums, the row sum being $s(x, y) = s(x, y - 1) + i(x, y)$ and then calculating the integral image value at (x, y) according to $ii(x, y) = ii(x-1, y)+s(x, y)$.

The integral image will quickly provide rectangle intensity sums for rectangles starting at the origin, but arbitrary rectangles also have to be computed. These rectangle sums can be calculated easily by means of the integral image as can be seen by the illustration below - the empty area on top of and left of the intended rectangle have to be subtracted from the large rectangle that spans from top-left to the current point whose area can easily be obtained from the integral image. Because these subtractions overlap we need to add a small top-left portion where the subtraction occurred twice. Therefore, only four array lookups are needed to calculate the intensity sum of a rectangle.

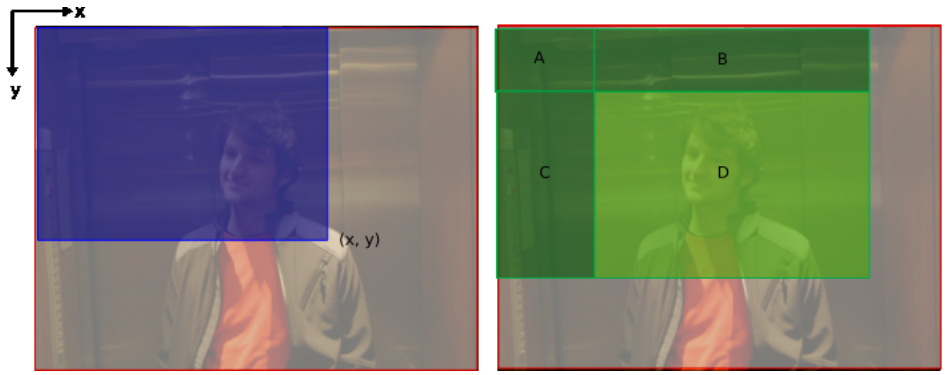


Figure 3: Integral image representation $ii(x; y)$ is the sum of all pixel intensities contained in the box from the upper-left corner to the current point. Computing the intensity sum of an arbitrary box is achieved by subtracting the sum of pixel intensities of boxes B and C from the main box and adding the sum of A, because A, B, C overlap.

2.1.3 Boosted classifier cascades

Even rather small search windows yield up to 180,000 possible feature combinations. Therefore an effective feature selection algorithm needs to select the most promising features. Viola and Jones [18] proposed to use a modified AdaBoost algorithm.

AdaBoost builds strong classifiers from so-called weak classifiers. It combines several of these weak classifiers via linear combination into a strong classifier that can fulfill a given classification task even though the weak classifiers it consists of need only be slightly better than chance. It does so by selecting the most discriminative features greedily. In the framework presented by Viola and Jones in [18] thresholded Haar-like features are used as weak classifiers. And according to [18] the great challenge is to find those features that can be combined to form a good classifier.

Consequently, those weak learners are chosen that separate positive and negative examples as good as possible. This is achieved by using a thresholded classification function that tries to minimize misclassifications of examples. Thus a weak classifier $h_j(x)$ consists of a feature f_j, a threshold θ_j and a parity sign p_j that indicates the direction of the inequality and x being a 24x24 search window:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Selecting many features will increase classification performance as many sample details can be modeled but at the same time many features lead to higher computation times. That results from the fact that in the current framework all features are evaluated over every possible search window in the image. To avoid evaluating all selected, discriminating features *cascades* are introduced. They are basically degenerate decision trees where at every stage a search window is either rejected or passed to the next stage, a window is finally accepted if it passes the whole cascade. The idea is to have early stages that discard many negative samples while still accepting all positive samples and to have more discriminative stages later that were trained on harder examples. Ideally many easy negative search windows are discarded early on and only hard to classify windows proceed further through the cascade. Therefore, the entire cascade will be evaluated in the rare case of positive samples and not for every search window while still retaining all of its discriminative power.

Cascade training is done accordingly. The first stage is usually trained with random non-object class negative samples and object class positive samples. The first stage tries to accept nearly 100% positives while rejecting as many negatives as possible. Then the next stage is only trained with the falsely accepted negatives that the first stage missed and this continues through all further stages. Therefore every stage faces a harder classification task than the stage before and thus is better at rejecting false positives while still trying to accept nearly all true positives. Now, the overall correct detection rate D and false detection rate F of a cascade with n stages can computed according to

$$D = \sum_{i=1}^{n} d_i \tag{3}$$

$$F = \sum_{i=1}^{n} f_i \tag{4}$$

where d_i is the correct detection rate of stage i and f_i is the false acceptance rate of stage i. individual stage false acceptance and correct acceptance rates can be calculated from (3) and (4). For every stage additional weak classifiers (thresholded Haar-like features) are added until the stage meets its desired rates. If the overall classifier fails to achieve the overall desired rates more stages are added.

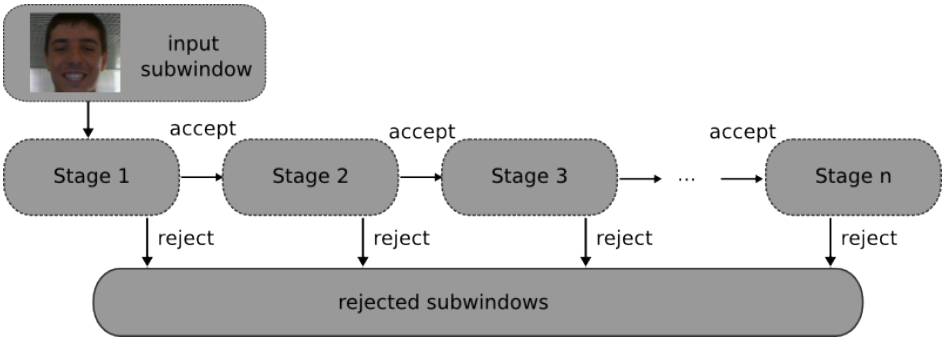


Figure 4: Haar classifier cascades.

2.2 Discrete cosine transformation

The discrete cosine transformation (DCT) is a frequency transformation of discrete-time signals, similar to the discrete Fourier transform. A difference is that the DCT works with real valued coefficients. It is widely used in signal processing and image compression because it has efficient implementations and provides compact data representation. Because most of the image information is usually concentrated in low-frequency components, it can be used to efficiently compress image data. It is almost as successful as the Karhunen-Loeve transform for image comrpression, in addition it uses data-independant bases and is faster to compute. The compression can be achieved by leaving out high-frequency coefficients and therefore smoothing the reconstructed signal.

DCT transforms a signal into a sum of cosinusoidal signals with different amplitudes and frequencies. As computer vision problems usually deal with images and therefore two-dimensional data, the 2-D DCT is defined as

$$F(u,v) = C(u)C(v)\frac{2}{X}\sum_{x=0}^{X-1}\sum_{y=0}^{Y-1} f(x,y)\cos\left(\frac{\pi u(2x+1)}{2X}\right)\cos\left(\frac{\pi v(2y+1)}{2Y}\right) \tag{5}$$

where $X \times Y$ is the input data size and $C(i)$ is defined as

$$C(i) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } i = 0 \\ 1 & \text{otherwise} \end{cases} \tag{6}$$

Figure 5 shows all precomputed DCT image bases for 8x8 input data size.

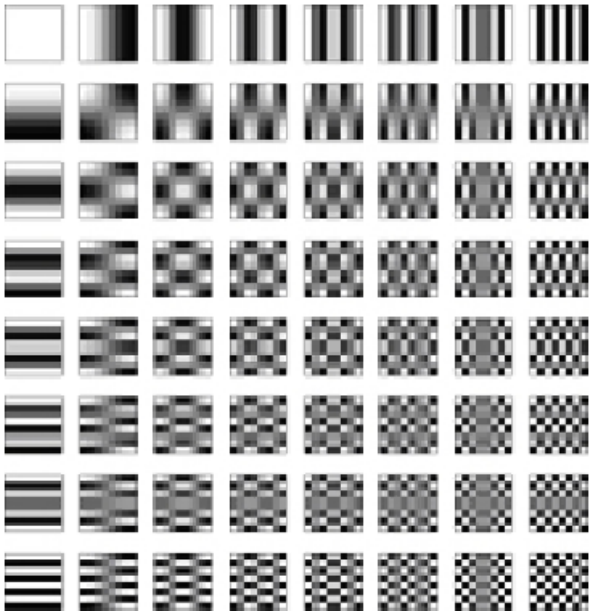


Figure 5: DCT image bases, width and height are 8 pixels.

2.3 Local appearance-based face recognition using DCT

Usually, one necessity for recognition is a transformation into a lower dimensional feature space because working with raw pixel intensities has several disadvantages. Classifiers trained on raw pixel intensity input are not robust against image noise and are strongly dependent on appearance, therefore large amounts of training data are required. The discrete cosine transformation (DCT) is such a transformation. One reason to prefer DCT over PCA is that DCT utilizes data-independent bases. As opposed to the subspace methods, the DCT bases do not have to be constructed from the training data first because the bases are predetermined. The effect of slightly misaligned training images for base construction is shown in [1], the resulting base images have more noise than those constructed from well-aligned training images. Therefore, alignment errors in the training data do not have a large impact on DCT-based classification performance. Even though discrete cosine transform employs data-independent bases, it approaches PCA's compact data representation ability where PCA is proved to be optimal. When DCT is used for feature extraction in a local appearance-based approach, recognition rates outperform those of holistic approaches [2]. Local appearance-based recognition using DCT splits an aligned face image into blocks of 8x8 pixels and applies the DCT transform on each block. Then, each block is reordered by zig-zag-scanning the coefficients (see Figure 10) and the top-left DCT coefficient of each block is discarded as it represents the block's average intensity value. Next, appropriate coefficients are selected in each block that yield the best classification results, in [2] it is shown that removing the first coefficient and selecting the following five provides good results. Then, fusion occurs either at feature level or at decision level - i.e. first concatenating the individual features into a single feature vector before classification or

classifying locally and then fusing the local decisions.

2.4 Support vector machines

Support vector machines (SVMs) are maximum margin binary classifiers that solve a classification task using a linear separating hyperplane in a high-dimensional projection-space. This hyperplane is chosen to maximize the distance between positive and negative samples. Real-world problems seldom present linearly separable data, therefore a transformation into a higher-dimensional space is applied before classification with hopes of being able to linearly separate data in the new space. The great advantage is that the hyperplane does not need not be projected down into the original space, instead the classification takes place right in the high-dimensional space implicitly. SVMs are conceptually simple yet powerful and the results are interpretable, all good reasons to employ SVMs. Further introduction into SVMs can be found in [14] and [15].

2.4.1 Linear classification

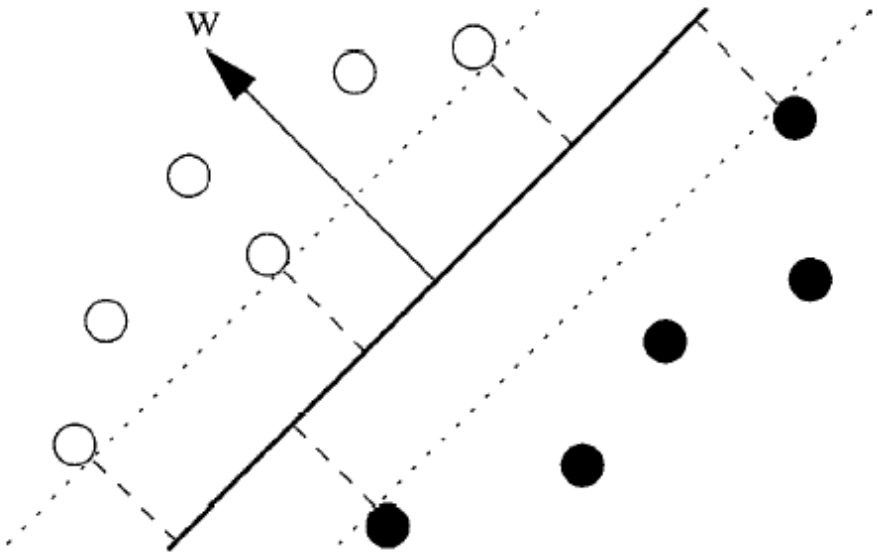


Figure 6: Linear classifier and margins - the margin is proportional to the expected generalization ability. Taken from [15].

Let $\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$ again denote the training data, consisting of a training vector x_i and a corresponding classification class $y_i \epsilon \{-1, 1\}$. Under the assumption that the provided data can be separated linearly in an n-dimensional space, we can construct an infinite amount of $n-1$-dimensional hyperplanes that correctly

separate the training data, because there are no restrictions on placement or orientation of the hyperplane as long as the data is correctly classified. The idea of maximum margin classification is then to choose the hyperplane with the maximum separating margin between the two classes because it can be expected that this maximum margin hyperplane is best at generalization, i.e. the margin is proportional to generalization ability of the classifier.

A hyperplane can be described as

$$\{x \epsilon \mathbb{S} : wx + b = 0, (w, b) \epsilon \mathbb{S} \times \mathbb{R}\} \tag{7}$$

and the maximum margin separating hyperplane has to minimize the condition $\min_{i=1\ldots n} |wx_i + b| = 1$ where x_i are the training examples. Consequently, the distance between two samples x_i and x_j relative to the hyperplane can be defined as $\frac{w \cdot (x_i - x_j)}{\|w\|}$. Then the distance between the two classes is $\frac{2}{\|w\|}$. So in order to classify training data correctly the hyperplane can be found by maximizing $\frac{2}{\|w\|}$ or minimizing $\|w\|^2$ under the condition

$$y_i(wx_i + b) \geq 1 \quad \text{for} \quad i = 1 \ldots n, \tag{8}$$

that guarantees that all samples are correctly classified. This minimization can be achieved using Lagrange multipliers once rewritten into

$$L_p = L(w, b, a) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i(wx_i + b) - 1)\,. \tag{9}$$

with $\alpha_1, \alpha_2, \ldots, \alpha_n$ being Lagrange multipliers. After solving the optimization problem most α_i are zero because their conditions are fulfilled. Those x_i whose $\alpha_i > 0$ are chosen as support vectors to represent the margins, they are the closest vectors to the hyperplane. Then w can be computed as a linear combination of these α_i: $w = \sum_{i=1}^{n} \alpha_i y_i x_i$.

2.4.2 Soft-margin linear classification

In order to allow a certain number of misclassifications a soft-margin is introduced. The optimization condition is changed to $y_i(wx_i + b) \geq 1 - \xi_i$ for $i = 1 \ldots n, \xi_i \geq 0$, where ξ_i is the sample x_i's distance from the correct margin, it is sometimes also referred to as slack term. Therefore if $\xi_i > 1$ the sample is misclassified, if $0 < \xi_i < 1$ the sample is correctly classified but within the margin (margin error) and if $\xi_i = 0$ the vector lies on the margin. Consequently, the minimization term becomes

$$\min_{w,b,\xi_i} \|w\|^2 + C \left(\sum_{i=1}^{n} \xi_i \right) \tag{10}$$

where C is a weighting parameter that controls the rate of misclassifications - small C maximizes the margin, large C yields few misclassifications.

So soft-margin classifiers allow a certain number of misclassifications and can therefore better cope with data that is not exactly linearly separable.

2.4.3 Non-linear classification

Given that most real-world data is not linearly separable, the data has to be altered in some way in order for the previous linear maximum-margin classifiers to be useful. The idea is to transform the data into a higher-dimensional space and scatter the data suitably so that it can then be classified using linear separation.

The transformation is usually of the form $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^m, m > n$. But transformations and computations in high-dimensional spaces are usually computationally expensive. Therefore the so-called "kernel-trick" is employed, a kernel function is defined as the dot product of the projection of two vectors - $K(x, y) = \Phi(x) \cdot \Phi(y)$. As the previous equations exclusively use dot products in the high-dimensional space there is no need to explicitly transform the data or to transform the hyperplane. Instead, all equations can be evaluated using kernel functions. Popular kernel functions are
Polynomial:

$$K(x, y) = (x \cdot y + c)^d \tag{11}$$

Radial basis functions:

$$K(x, y) = \exp \frac{-\|x - y\|^2}{2\sigma^2} \tag{12}$$

Sigmoid:

$$K(x, y) = \tanh(\kappa(x \cdot y) + \theta) \tag{13}$$

c, d, σ, κ and θ are parameters and have to be chosen to optimize the classification.

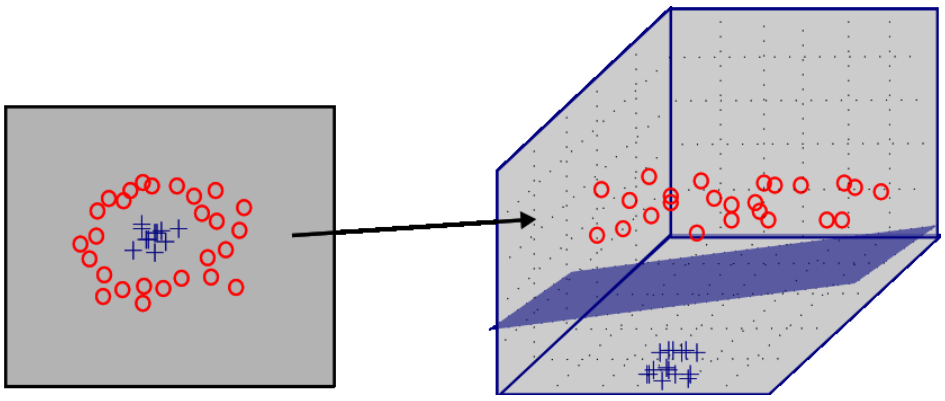


Figure 7: Non-linear classification using kernel function. A kernel function is used to transform data into a higher-dimensional space where the data is linearly separable.

2.5 Nearest neighbor classification

Nearest-neighbor (NN) classification is a delayed or lazy learning algorithm. It employs simple instance-based learning where the learning itself consists of storing the training data samples. The actual work is done later during classification. This makes NN classification easy to implement and non-parametric. On rather small data sets NN

classification is on par computationally with other classifiers and may even outperform complex classification algorithms.

Given the training data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$, consisting of a training vector x_i and a corresponding classification class y_i, and a vector x that is to be classified, NN is defined as

$$c = \underset{i\epsilon 1,\ldots,m}{\operatorname{argmin}} \|x - x_i\| \tag{14}$$

.

The result is the class of sample x_c, namely y_c. This approach can easily be extended to k nearest neighbors. In that case, the decision is made by combining the each nearest neighbor's decision in some way.

Usually, nearest-neighbor classification is performed in an Euclidean space using the Euclidean distance metric, also known as L2 norm. Other norms may be used, such as the L1 or city-block distance. It is defined as

$$d(x_1, x_2) = \sum_{b=1}^{N} |x_1[b] - x_2[b]| \tag{15}$$

where $x_i[b]$ is the bth component of the vector x_i and N is the length of vectors x_i.

A detailed evaluation of the use of different distance metrics in nearest-neighbor classification for local appearance-based face recognition is performed in [2]. The paper states that usually the L1-norm outperforms other norms. Therefore we use L1-norm in nearest-neighbor classification later in this work.

2.6 K-Means clustering

The k-means algorithm is an unsupervised learning algorithm that is used to cluster data, i.e. classify objects into different categories, into k partitions. It was introduced by MacQueen [16] and is related to the expectation-maximization algorithm. In the k-means approach clusters are represented by their respective centroid or mean. Due to its conceptual simplicity and efficiency it is used widely.

K-means tries to reduce the sum-squared distance of all data-points x_j and its associated cluster mean μ_i. The error term can be formulated as

$$SSE = \sum_{i=1}^{k} \sum_{x_j \in S_i} d(x_j, \mu_i)^2 \tag{16}$$

where $d(x_1, x_2)$ denotes the distance between a data-point and a cluster-mean. Generally, Euclidian and city block distance are used as the distance metric.

The simplest form of the algorithm starts by selecting k random cluster means, e.g. by selecting k random data points as initial cluster means. Then, every data point is assigned to its nearest centroid according to the distance function. The next step is to update the centroid locations by calculating the mean of all assigned data points to each current cluster. These steps are iterated until either the centroids' values no

longer change, until the data points do not switch between clusters or until a certain number of iterations is reached.

K-means may in fact converge to local minima. One solution to avoid only locally optimal solutions is to start with different random centroids and run the algorithm several times.

3 Open-Set Face Recognition

This chapter covers the implementation and is organized as follows. Section 3.1 describes how the face registration component works, Section 3.2 explains the feature extraction procedure. Then, Section 3.3 introduces the classifiers that were employed and Section 3.4 provides information about the use of additional video-based information.

3.1 Face registration

Face registration is the task of aligning the faces such that different faces are transformed to a common coordinate system. This task is a crucial preparation step for face recognition. Since most recognition algorithms are quite sensitive to even small changes in orientation or position correct registration is very important. If registration fails recognition cannot be performed successfully.

Face registration can be considered as a normalization step and consists of two phases. First, a face detection algorithm needs to find the face. Then, an eye detection algorithm is used to detect the eyes. The eye locations are used as reference points, and the images are transformed so that eye locations in the registered images are the same.

3.1.1 Face detection

Face detection is done by using Haar cascade classifiers. The classifiers were trained on several face and randomly chosen non-face images. The detected outline is usually not suited to be taken directly for recognition. The final Haar cascade classifier output is a fusion of multiple detections that can occur at different scales and slightly different positions for same face, so it can be rather crude. Therefore further steps need to be taken.

In order to be able to handle multiple persons appearing and disappearing, in case of multiple face detections the largest hit with the least distance from the last detection is selected.

3.1.2 Eye detection

As stated above, face detection is just a rough first step to estimate face position. The returned rectangle can vary significantly in size and sometimes location, a first detection may cover just the skin-colored face area, whereas the next detection may cover the whole face including neck and hair. It is difficult to establish robust recognition if the variation between two consequent detections of the same person may already be larger than the overall differences between two totally different people. Therefore, in addition to face detection, eyes are detected as well to provide more robust cues about face orientation and size.

Eye detection is also done using Haar cascade classifiers. Eyes seem to pose a greater problem than faces for the Haar cascade classifiers because they contain less discrimi-

natory intensity features. Fortunately, once a face is detected there are not too many other similar facial features. Nonetheless, false positive detections may occur around the nose or far more often e.g. on curly hair. Haar cascade detection may fail completely if the eye-brows are covered by hair. That is because just the eyeball's appearance by itself contains too few discriminatory information to robustly detect eyes and therefore additionally eyebrows were included in training images as there is usually a high contrast difference between the eye brows and the surrounding area, which improves the detector.

Additionally, to improve robustness a few checks were included. The eye detection has to be within the upper half of the face, the left eye must be within the left upper half, and the right eye within the right upper half of the face. In order to minimize the jitter of eye detections the newly detected eye coordinates are compared with the last detection and the closest hit is chosen.

If the eye detector fails to detect either eye an educated guess is made based on the current face location and old eye positions. This ensures that infrequent mismatches can be compensated.

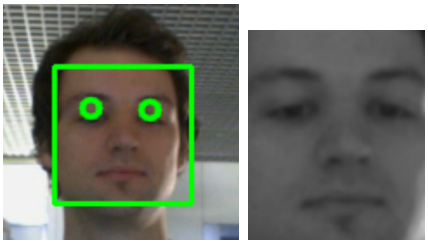
3.1.3 Alignment

After face and eye positions have been established alignment is straight forward. The image is rotated and then cropped according to the distance between the eyes. The output of the aligner is a cropped and rotated face image of 128x150 pixels size, the distance between eyes is 70 pixels and the eyes are located 45 pixels from the top of the image. The resulting registered image is then scaled to 64x64 pixels size, the eye locations in registered images are the same.

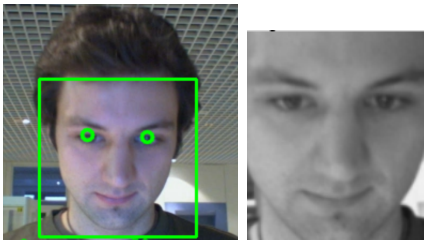
3.1.4 Sample augmentation

Imperfect registration may decrease recognition performance but perfect registration is very hard to achieve and not always possible. The advantage of creating additional samples is that a classifier trained with these samples becomes more robust against registration variations. This is important because registration in real-world applications is usually not perfect. In order to overcome this difficulty an additional processing step is performed. Before applying the DCT transform to an aligned and cropped face image a set of related images are generated by slightly varying detected eye locations and therefore creating new images with slightly different alignment.

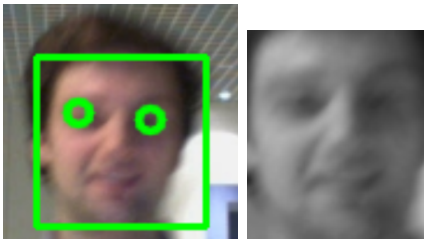
In order to simulate the influence of imprecise registration, each eye's location is varied individually by a few pixels in each direction, for example in the 4-neighborhood of the detection. A variation of the eye location by a single pixel in each direction ($\{-1, ..., 1\}$ in x direction and $\{-1, ..., 1\}$ in y direction per eye) yields 24 additional samples. Consequently, 25 times more registered face images are available with just a variation of a single pixel. Increasing the amount of data may cause problems, it may lead to sample data imbalances. So, after creating these additional samples and extracting the features k-means clustering is applied to reduce the amount of positive samples.



(a) Daylight



(b) Artificial light, head pose variation



(c) Daylight, motion blur

Figure 8: Face and eye detections

Figure 9: Face with failed eye detection

K-means clusters the samples into k clusters represented by their means and these cluster means may then be used as the representative samples.

The approach is therefore to create additional samples by varying eye detections in the 4-neighborhood of the original detection, this step yields 24 additional samples per original training image. Then, every sample is divided into blocks, discrete cosine transformed and the relevant coefficients are extracted to form a feature vector as described in Section 3.2. Each feature vector represents a point in the feature space so the standard k-means algorithm can be applied to the set of all extracted features for each client individually. In order to reduce the size of additional samples to the initial

amount of data, the clustering parameter k is chosen as $\frac{N}{25}$, where N is the number of initial samples and clustering is done using the L1 distance metric. After clustering, the usual training and classification steps can be performed.

3.2 Feature extraction

Every cropped and aligned face image of 64x64 pixels resolution may be interpreted as a single point in a 64x64 dimensional space. But too many training examples would be needed to perform classification in such a high-dimensional space. Therefore the dimensionality of the features has to be reduced. The standard approaches to dimensionality reduction include principle component analysis (PCA), linear discriminant analysis (LDA) and discrete cosine transformation (DCT). DCT has the advantage of utilizing data-independent bases which limit the influence of registration errors because these bases do not have to be constructed from automatically registered input images during training. Misaligned training images decrease the performance of subspace methods because noise is introduced in the basis images, see [2] for examples. OpenCV was used to extract DCT features.

3.2.1 Local DCT coefficients

Every aligned and cropped face image to be classified is first converted into a grayscale image of 64x64 pixels and then divided into 8x8 pixel blocks, then DCT is performed on each of these local blocks. A block size of 8x8 pixels provides a reasonable compromise between compression and processing overhead.

After dividing the image into 8x8 blocks DCT is performed on each block. Then, each block's coefficients are ordered using zig-zag-scanning, see Figure 10. From that sorted list, the first, so-called DC, component is skipped because it represents the average pixel intensity of the entire block. The following five coefficients are retained and the remaining coefficients are discarded. Especially these low-frequency coefficients have been shown to perform well as features for classification tasks. The process then yields five DCT coefficients for every image block. There are in total local 64 blocks in 64x64 input images, so the dimensionality is reduced from 4096 to 320.

3.2.2 Feature normalization

DCT preserves the total image energy of the processed input block, therefore blocks with different brightness levels lead to DCT coefficient with different magnitude values. In order to balance each local block's contribution to the classification the local feature vector is normalized to unit norm. The magnitude of feature vector f is transformed into the normalized feature vector f_n:

$$f_n = \frac{f}{||f||} \tag{17}$$

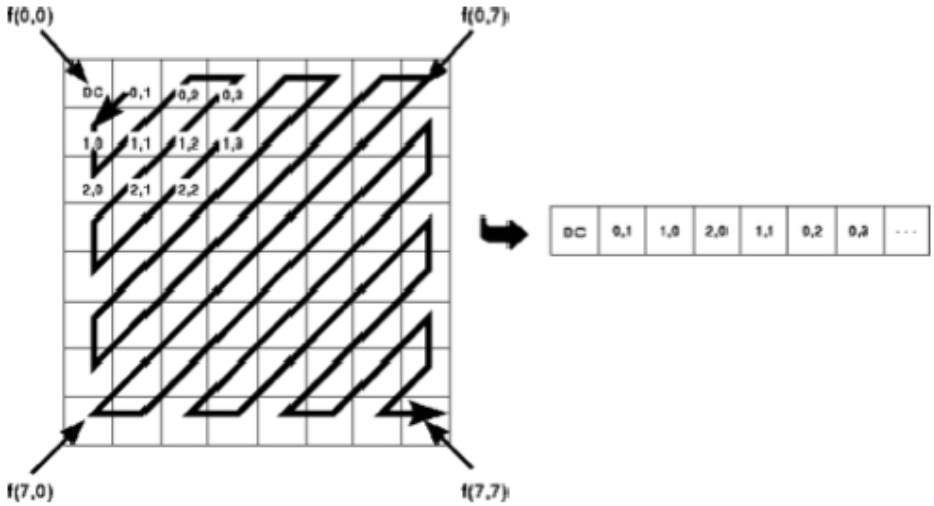


Figure 10: Zig-zag scanning coefficients and construction of the DCT feature vector.

3.3 Classification

Both support vector machine and nearest-neighbor classifiers are used to perform the open-set classification task. LibSVM was used as the SVM implementation.

When classifying a known person in open-set recognition, the system may either correctly accept the person as known and identify him or her correctly, correctly accept but falsely classify the person or the person may be falsely rejected. An unknown person may be correctly rejected or falsely accepted. Taking these into account, the performance measures are defined as: CCR (correct classification rate, correct acceptance and identification of known), CRR (correct rejection rate, correct rejection of unknown), FAR (false acceptance rate, false acceptance of unknown), FRR (false rejection rate, false rejection of known) and FCR (false classification rate, correct acceptance but misclassification of known).

The error rates are defined as

$$FAR = \frac{n_{impostor,accepted}}{n_{impostor}} \tag{18}$$

$$FCR = \frac{n_{genuine,misclassified}}{n_{genuine}} \tag{19}$$

$$FRR = \frac{n_{genuine,rejected}}{n_{genuine}} \tag{20}$$

where $n_{genuine}$ and $n_{impostor}$ represent the number of genuine and impostor samples presented to the recognition system.

3.3.1 SVM based classification

Open-set recognition is a multi-class classification problem, support vector machines on the other hand are binary classifiers. Traditionally, two ways were employed to solve multi-class problems with SVMs, either a one-vs-one or one-vs-all approach.

The one-vs-one solution works by training $\frac{m(m-1)}{2}$ classifiers to solve an m-class problem. Each classifier is trained to separate two single classes, this is a pairwise approach.

In the one-vs-all approach m classifiers are trained to solve the m-class problem. Each SVM separates one class from all other remaining classes.

This work formulates the open-set recognition problem as a multi-verification task. As described by McKenna et al. in [8], open-set face recognition can be formulated as a series of 2-class verification problems. Given a claimed identity, the result of an identity verification is whether the claimed identity is true or false. Given a number of positive and negative samples it is possible to train a classifier that models the distribution of faces for both cases. In order to carry out open-set recognition, an identity verifier is trained for every one of the n known subjects in the database. Then in order to perform open-set recognition, the probe is presented to each verifier and n verifications are performed. Once a new probe is presented to the system, it is checked against all classifiers, if all of them reject, the person is reported as unknown; if one accepts, the person is accepted with that identity, if more than a single verifier accepts, the identity with the highest score wins. Scores are inversely proportional to the nearest-neighbor distance. For the n-best list creation, all distances are converted into scores and then a min-max normalization is performed on the list.

The two-class nature of SVM classification lends itself to formulating the open-set recognition task as a verification problem. Although one-class SVMs exist they were not used because there is a sufficiently large set of training data that can be used to model the class of unknown individuals. Providing samples for both classes allows the SVM classifier to choose the optimal separating support vectors and therefore provide a better model of the probability density function than a one-class model that has to estimate the boundary without actually having negative examples.

Since this work focuses on video-based classification, n-best lists are used instead of a single frame-based score as will be explained later. The advantage of using n-best lists is that no hard decisions are made immediately so that initial false decisions may be revised later. N-best lists are lists containing the n best classification results ordered by classification score. Therefore scores need to be derived, in case of SVM classification the score is the distance from the hyperplane. Unlike calibrated sigmoid posterior probabilities as proposed by Platt in [24] these do not represent posterior probabilities. In order to reduce variation of distances between frames, the score is *min-max-normalized* [25] as

$$s'_i = 1 - \frac{s_i - s_{min}}{s_{max} - s_{min}} \qquad i = 1, 2, ..., n. \tag{21}$$

This maps scores to $[0, 1]$ and then all scores are renormalized to yield $\sum_{i=1}^{n} s'_i = 1$ so that each frame has an equal contribution.

Training data imbalance consideration

A large set of negative samples, i.e. training samples of unknown clients, is necessary to capture the large variance of the unknown class in the multi-verification approach. Every recording session consists of roughly 150 frames, unknown people have a single recording session which is used for training, whereas four sessions are used for known training. In order to have balanced data, i.e. an equal number of positive (known) and negative (unknown) training images we would need to have as many recordings of the single known person as from all unknowns combined. Therefore, if we use four session per known for training with ≈ 600 frames against 20 unknown individuals with ≈ 150 frames each (≈ 3000 total) we would have a 1:5 imbalance. In case of only a single available session per known the imbalance is 1:20.

This problem was tackled by undersampling the unknown sequences here. Undersampling works by selecting only every nth frame to train the classifier if there is an imbalance of 1:n.

As Akbani et al. state in [13] undersampling shows a significant performance gain. Undersampling is inferior to the proposed SDC algorithm presented in [13] but outperforms pure SVM classification on unbalanced datasets.

3.3.2 Nearest-neighbor based classification

The nearest-neighbor based classification method uses the same sort of multi-verification approach. Again, for every person to be recognized an individual classifier is trained.

The problem is that the samples of some subjects may have a larger variance in distance than the samples of some other individual. Nonetheless, we would like to use a common threshold that can be tuned globally - for example by finding the equal error rate.

Each nearest-neighbor classifier calculates a normalized distance, this normalized distance is simply shifted by the mean and divided by the standard deviation that is based on the distances obtained on the training samples:

$$d' = \frac{d - \mu_{c_i}}{\sigma_{c_i}} \tag{22}$$

where d is the distance of the classifier's input vector to the nearest training sample in class c_i and μ_{c_i} and σ_{c_i} are mean and variance of distances of class c_i estimated using the training data. If this normalized distance is lower than the global threshold the probe sample is accepted by the verifier.

3.4 Video information incorporation

This section explains how additional information from video sequences is used to improve classification performance.

3.4.1 Temporal fusion with progressive scores

Since a person's identity does not change during the video capture, one may try to enforce temporal consistency. In order to make it possible to revise a preliminary decision later on, instead of relying on a single classification result for every frame an n-best list is used. N-best lists store the first n highest ranked results, n may be chosen freely, $n = 3$ was used in this work. Then, for each recognized identity a cumulated score is stored that develops over time.

First, the frame n-best list is computed as described in Section 3.3.1 - scores of all accepting classifiers are gathered into a list, that list is min-max-normalized and then renormalized so that the sum of all entries is 1. Then, the scores of the current frame are added to sequence scores. Then, a decision may be made based on sequence scores. Given enough time the real identity should accumulate the highest score because it should have highest individual frame scores and should have most acceptances.

If there is no face detection over multiple frames in the live system the cumulated scores are reset because it is assumed that the person has left. Resetting scores allows the whole process to restart once another person is located.

4 Experiments

This section presents the data set and the experiments that were conducted on the data.

The following section is organized as follows: first the data set is described, then the open-set face recognition performance of both nearest neighbor and SVM-based baseline systems is compared. After evaluating these baseline systems further optimizations are introduced and evaluated.

4.1 Experimental setup

The experiments were conducted on a data set that was recorded between January and May 2007. The data set consists of real-world data collected in a wide hallway in front of an office. The system consists of a portable laptop computer and a connected webcam that was used to capture short video sequences of the subjects.

The following experiments were performed on an implementation based on LibSVM for SVM-based classification. The SVM parameters were chosen by performing a simple grid search to find the optimal combination of parameters that yield the best results on an independent cross-validation set based on the FRGC data. The parameter values are listed in Table 1.

SVM parameters	
SVM type	C-SVM classifcation
SVM kernel	Polynomial $(\gamma x_i x_j + \text{coef0})^d$
Polynomial degree d	2
γ	2
C	32
coef0	0
ϵ	10^{-10}

Table 1: SVM parameters

The nearest-neighbor-based classification was performed using the L1 distance metric as supported by results from Ekenel and Stiefelhagen [2].

4.2 Data set

55 people were recorded in front of the office. Lighting was natural or artificial, depending on the time of the day. These recordings were split into two groups, a group of known people and a group of unknown people. From now on, we will use the term **known** for those subjects that are added to the database during training, i.e. those probes that have mates in the gallery (or database), **unknown** will refer to subjects who are not present in the gallery.

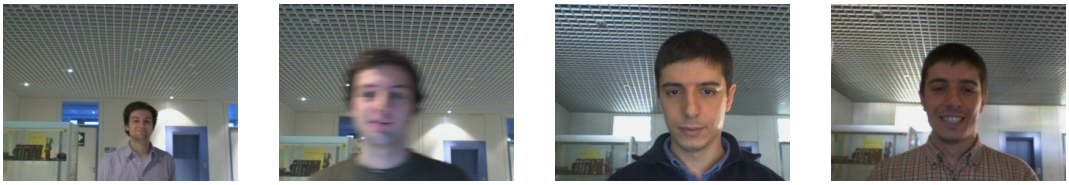
Each recording session consists of approximately 150 registered frames, i.e. frames in which both face and eyes were detected. The recordings are split into training and testing data. Recordings were executed over a series of four months, so some sessions are months apart.

Different sets of data were used for training and testing. Known people's recording were split into training and testing sessions which do not overlap. Unknown subjects used for training are different from those used for testing.

Different experiments required slightly different partitionings of the data, please refer to each experiment for a short explanation of the corresponding data set. Depending on the experiment, either two or four training sessions were used to train known people. 25 clients' sessions were used as unknown training data, the remaining known sessions and 20 different unknown clients were used for testing.

Data set	
# sessions per client	# clients
1 session	36 clients
2 sessions	4 clients
3 sessions	7 clients
4 sessions	2 clients
5 sessions	1 client
7 sessions	3 clients
8 sessions	1 client
11 sessions	1 client

Table 2: The data set



(a) Artificial light, far away (b) Artificial light, motion blur (c) Daylight, brighter (d) Daylight, darker

Figure 11: Recordings from the data set, different illumination and face sizes

4.3 Open-set face recognition performance

In this subsection, the open-set identification performance of the proposed system is analyzed. SVM-based and nearest-neighbor-based classification approaches are compared and contributions of error types are analyzed based on a baseline system.

4.3.1 Comparison of baseline SVM-based and NN-based open-set face recognition

The types of errors that can occur during open-set recognition are explained in Section 3.3 and defined in Equations 18, 19, 20.

In open-set recognition three error terms, namely FAR, FRR and FCR, have to be traded off against each other and it is not possible to minimize them at the same time. Therefore the equal error rate (EER) performance measure is employed as a combined error rate measure.

The EER can be found by choosing a threshold for which

$$FAR = FRR + FCR. \tag{23}$$

Support vector machines automatically minimize the overall error and try to find the global minimum. Therefore if the decision hyperplane is not altered these errors are all automatically minimized. Nonetheless, it may be desirable to fine-tune the system. Depending on the intended use of the system - either for pure recognition or secure applications - it may be desirable to reduce the number of false rejections and false classifications or the number of false acceptances. The ROC curve for SVM-based classification was created by using a parametrized decision surface as introduced by Phillips [22]. The decision hyperplane $\{x \epsilon \mathbb{S} : wx + b = 0, (w, b) \epsilon \mathbb{S} \times \mathbb{R}\}$ is modified to $wx + b = \Delta$, where Δ allows to adjust the false acceptance rate and the correct classification rate accordingly.

Nearest-neighbor-based classification does not automatically yield the best performance, a good threshold value has to be selected first. As the nearest-neighbor-based classification uses normalized distances a global threshold can be used, the threshold defines the maximum distance of a test sample to the closest stored training sample of a given class up to where this test sample is accepted as known. This global threshold is chosen to satisfy EER criterion.

4.3.2 Performance comparison

This subsection compares open-set classification performance of nearest-neighbor and SVM-based classification. The classification results reported in Table 4 and Table 5 were obtained using the data explained in Table 3 and the baseline algorithm using frame-based classification and undersampling.

Nearest-neighbor-based classification

Training data		
Known	5 subjects	4 sessions and ≈ 600 frames per person
Unknown	25 subjects	1 session, 30 frames per subject
Testing data		
Known	5 subjects	3 – 7 sessions per person and 3646 frames overall
Unknown	20 subjects	1 session per person and 3563 frames overall

Table 3: Data set for open-set performance experiments

Classification	CCR	FRR	FAR	CRR	FCR
Frame-based	81.6 %	15.5 %	17.8 %	82.2 %	2.9 %

Table 4: Nearest-neighbor classification results

The above results were obtained using a global threshold of 2.37 that was chosen using the equal error rate criterion as in Equation 23.

SVM-based classification

Classification	CCR	FRR	FAR	CRR	FCR
Frame-based	90.9 %	8.6 %	8.5 %	91.5 %	0.5 %

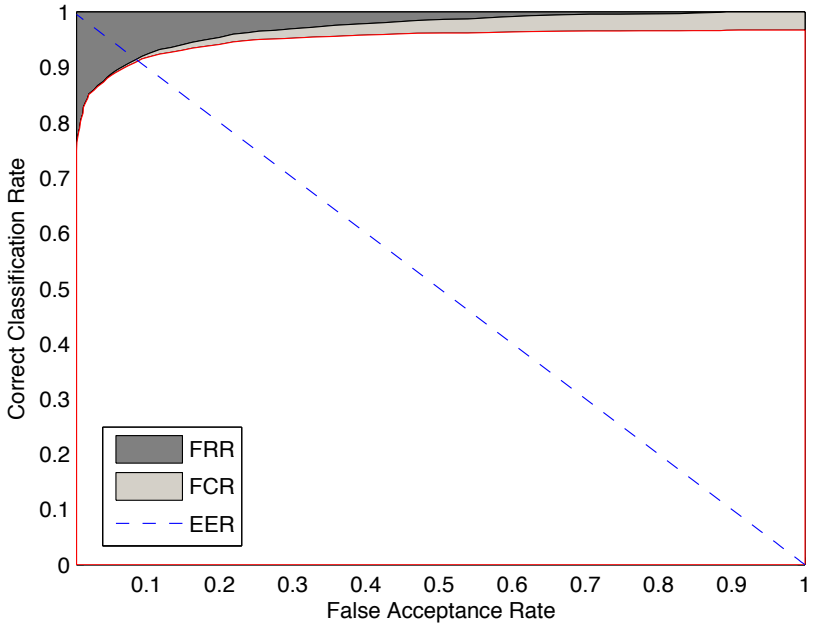
Table 5: SVM classification results, with parametrized hyperplane $\Delta = -0.12$

The SVM-based classification results were calculated using $\Delta = -0.12$ for the parametric hyperplane, according to EER criterion.

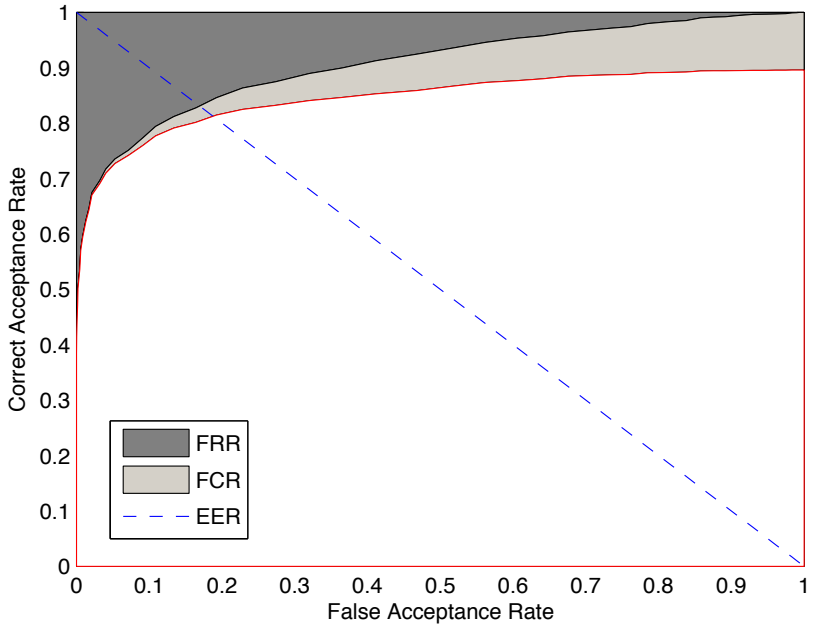
Table 4 and Table 5 show that SVM-based classification outperforms nearest-neighbor classification. Studies like [17] suggest that support vector machines perform well in extracting relevant discriminatory information. The nearest-neighbor-based classification could be further optimized by employing client-specific thresholds and determining individual thresholds by means of the EER criterion for each client.

Receiver Operating Characteristic (ROC) curves for both SVM- and nearest-neighbor-based classification are given in Figure 12. The ROC curve plots the correct classification rate against the percentage of impostors accepted by the system. In Figure 12, the FAR is represented as the value on the x-axis and correct classification rate is plotted on the y-axis.

Figure 12 a) and b) allows to examine the errors made during classification more closely. The individual contributions of the FRR and FCR can be seen from the graphs. Figure 12 shows that at the point of equal error the SVM-based classifier outperforms the NN-based classifier as the CCR is higher.



(a) SVM Receiver Operating Characteristics curve



(b) Nearest-neighbor Receiver Operating Characteristics curve

Figure 12: Receiver Operating Characteristics curves

4.4 Further experiments

This section describes further experiments that were conducted to analyze the performance of the proposed system. Unless otherwise stated, SVM-based classification was performed using the frame-based baseline approach without optimizations, the hyperplane parameter was kept at $\delta = 0$.

4.4.1 Influence of the number of training sessions

The number of training sessions has an influence on the classification results. The more training sessions are used the more likely is a good coverage of different poses and lighting conditions. This results in a better client model with more correct acceptances and less false rejections.

The results in Table 6 were generated using frame-based classification without any progressive accumulation of scores. The data set used is described in Table 3. Four sessions were set aside for training of which not all were used depending on the desired amount of training sessions. Therefore multiple combinations of training sessions are possible if less than the maximum of four sessions are used for training. In order to obtain meaningful results, all possible combinations were generated - 1024 for a single training session, 7776 and 1024 for two and three training sessions and only a single combination for four training sessions per person. Of these, thirty combinations, if available, were randomly selected for every run. Table 6 reports the mean (μ) of the classification rates and their standard deviation (σ) that were computed from the results of each of these thirty combinations for every run.

Table 6 and Figure 4.4.1 show that the classification results improve when more data is available. Figure 4.4.1 plots average classification rates and standard deviations for different number of training sessions. More training data covers a greater range of variations of the subject in pose, illumination and expression and allows the classifier to better model the subject.

Training sessions	μCCR	σCCR	μFRR	σFRR	μFAR	σFAR	μCRR	σCRR	μFCR	σFCR
1 sess.	29.0 %	8.6 %	71.0 %	8.6 %	0.3 %	0.2 %	99.7 %	0.2 %	0.1 %	0.1 %
2 sess.	57.0 %	12.1 %	42.8 %	12.1 %	1.5 %	1.2 %	98.5 %	1.2 %	0.2 %	0.1 %
3 sess.	73.8 %	7.8 %	25.8 %	7.8 %	2.6 %	1.1 %	97.4 %	1.1 %	0.4 %	0.1 %
4 sess.	87.2 %	0.0%	12.5 %	0.0%	3.7 %	0.0%	96.3 %	0.0%	0.3 %	0.0%

Table 6: Influence of the number of training sessions

4.4.2 Influence of the number of subjects

In order to explore the influence of the number of known subjects in the training database the following tests were conducted. Again, frame-based classification was used and the results can be boosted by using progressive scores.

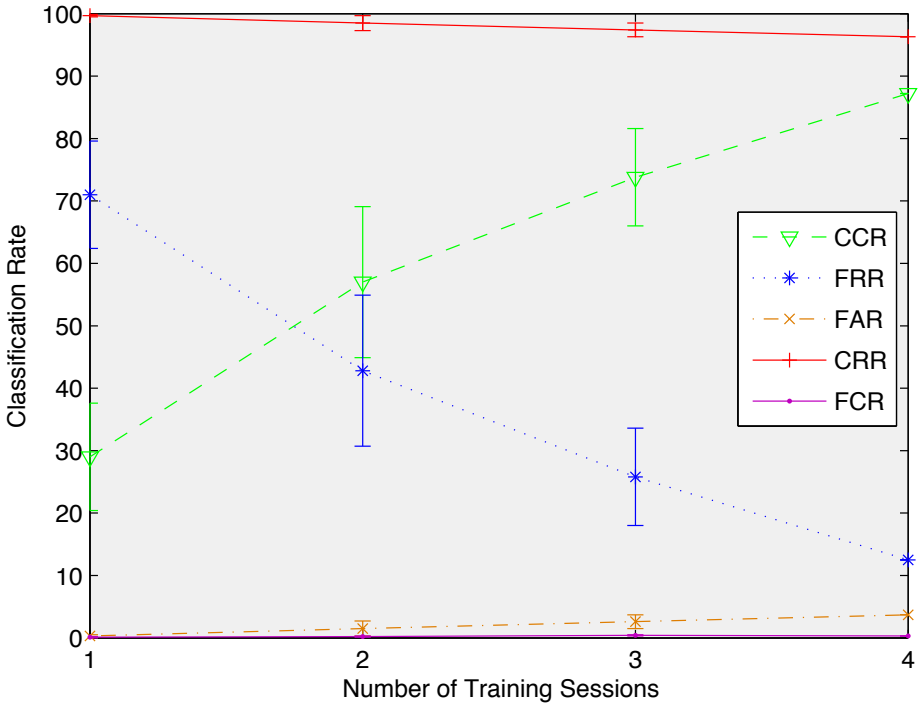


Figure 13: Development of classification scores with number of training sessions. See Table 6.

In order to evaluate the influence of the number of known subjects the system can recognize additional data had to be collected.

Training data		
Known	up to 9 subjects	4 sessions per person
Unknown	25 subjects	1 session, 30 frames per subject
Testing data		
Known	up to 9 subjects	2 sessions per person
Unknown	20 subjects	1 session per person

Table 7: Data set for testing the influence of the number of subject

Again, a fixed amount of training data was set aside for training. This time not the number of sessions was varied but the number of known subjects in the database. In order to generate meaningful results, again all possible combinations of known clients were generated for every experiment run. Here, all runs were evaluated and a total of 511 different training combinations were examined, Table 8 details the number of generated combinations. As in the previous section, Table 9 reports the mean (μ) of the classification rates and their standard deviation (σ) that were computed from the

results of each of these combinations for every run.

Due to limited available data only tests with up to nine known subjects could be performed. Security applications on the other hand have to recognize hundreds of known people but that is not the objective of this work. Table 7 explains the data set that was used in this experiment. Figure 4.4.2 displays the change in classification performance as more known subjects are added to the gallery. Table 9 presents the same results in numeric form and shows that CCR is mostly consistent on average with up to nine known subjects, the CRR decreases slightly as more subjects are added.

# Known subjects	# combinations
1 known id	9
2 known ids	36
3 known ids	84
4 known ids	126
5 known ids	126
6 known ids	84
7 known ids	36
8 known ids	9
9 known ids	1
Overall	511

Table 8: Number of examined combinations

Subjects	μCCR	σCCR	μFRR	σFRR	μFAR	σFAR	μCRR	σCRR	μFCR	σFCR
1 subj.	80.9 %	19.4 %	19.1 %	19.4 %	1.1 %	0.9 %	98.9 %	0.9 %	0.0 %	0.0 %
2 subj.	80.4 %	12.9 %	19.4 %	12.8 %	2.1 %	1.2 %	97.9 %	1.2 %	0.2 %	0.5 %
3 subj.	80.5 %	9.6 %	19.1 %	9.5 %	3.1 %	1.4 %	96.9 %	1.4 %	0.4 %	0.6 %
4 subj.	80.8 %	7.5 %	18.6 %	7.3 %	4.0 %	1.4 %	96.0 %	1.4 %	0.6 %	0.6 %
5 subj.	80.4 %	6.0 %	18.9 %	5.8 %	5.2 %	1.4 %	94.8 %	1.4 %	0.7 %	0.6 %
6 subj.	81.1 %	4.8 %	18.0 %	4.6 %	6.0 %	1.4 %	94.0 %	1.4 %	1.0 %	0.5 %
7 subj.	80.7 %	2.9 %	18.8 %	2.6 %	7.9 %	0.3 %	92.1 %	0.3 %	0.6 %	0.3 %
8 subj.	79.6 %	0.5 %	18.9 %	0.4 %	8.4 %	0.4 %	91.6 %	0.4 %	1.6 %	0.1 %
9 subj.	80.6 %	0.0 %	17.8 %	0.0 %	9.1 %	0.0 %	90.9 %	0.0 %	1.5 %	0.0 %

Table 9: Influence of the number of training sessions

4.4.3 Influence of time span between training and testing

As data was recorded over a time span of four months another interesting experiment would be to assess the influence of the period of time between training and testing on the performance. For most knowns there were different sessions which were recorded

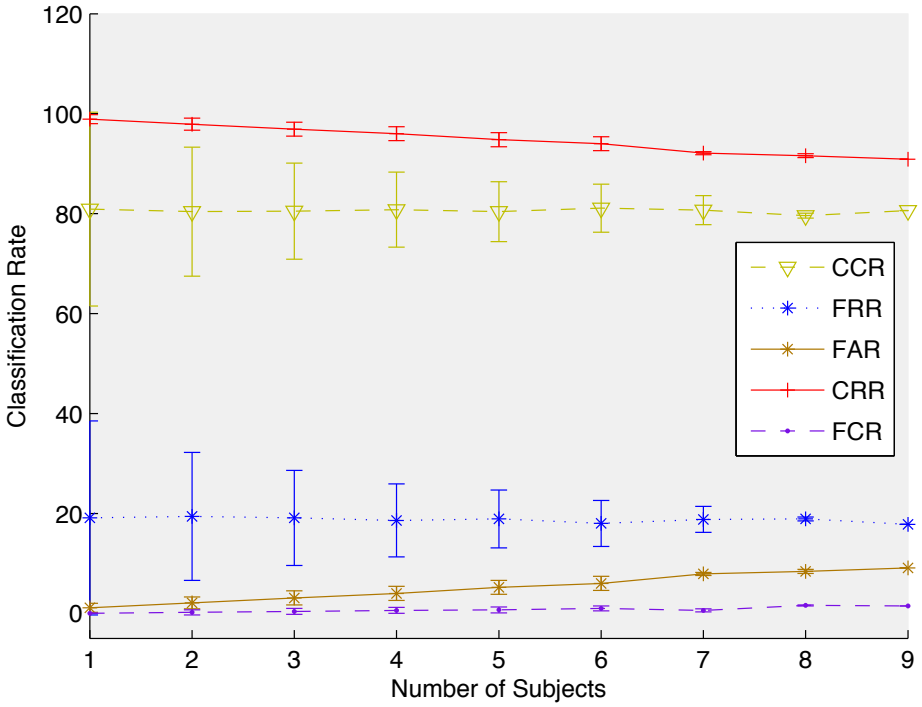


Figure 14: Development of classification scores with number of subjects. See Table 9.

up to three days after the first recordings, around ten days after initial recordings and around three months after initial recordings.

Table 10 shows that performance does not seem to degrade as the period of time between training and testing is increased. Due to the limited amount of available data there is variation in the results, the 10-day scenario performs better than the other scenarios. Overall it seems as if the age of the data (if less than three months) does not seem to degrade performance much. This is reflected by impressions from "live" testing sessions where old training data was used and the known person was still recognized from live video images some time later. This experiment suggests that a time span of three months between training and testing works well.

Time gap training-testing	CCR	FRR	FAR	CRR	FCR
$<$ 3 days	63.7 %	36.1 %	0.3 %	99.7 %	0.2 %
$\approx$ 10 days	77.2 %	22.8 %	0.5 %	99.5 %	0 %
$\approx$ 3 months	65.3 %	34.7 %	0.8 %	99.2 %	0 %

Table 10: Influence of time span between training and testing

4.4.4 Influence of undersampling

Akbani et al. [13] apply undersampling to different imbalanced data sets and show a performance gain. In this work undersampling was chosen as a simple way to avoid large imbalances.

The following experiment was conducted with the data set configuration explained in Table 3. Undersampling the unknown training data to 30 frames per session provides a balanced ratio with roughly 600 positives versus 750 negatives by using only every nth unknown frame, where $n = \frac{N}{n_u}$ and N is the total number of frames in the current unknown training session and n_u is the desired number of frames to be used for training. The results in Table 11 indicate that undersampling does improve classification performance.

# frames	CCR	FRR	FAR	CRR	FCR
30	87.2 %	12.5 %	3.7 %	96.3 %	0.3 %
60	85.2 %	14.7 %	2.7 %	97.3 %	0.1 %
90	83.5 %	16.5 %	2.4 %	97.6 %	0.0 %
150	83.5 %	16.5 %	2.3 %	97.7 %	0.0 %

Table 11: Effect of undersampling. Originally, 150 frames were available.

4.5 Classification performance optimization

This subsection explores possibilities to improve the classification performance of the simple frame-based baseline system.

4.5.1 Improvement of video-based classification

The availability of video data allows classification to be performed in three different ways, frame-based, progressive-score-based and video-based. In the frame-based approach every single frame is classified by itself, uninfluenced by other preceding frames. Therefore these results return some insight on the general performance of the classification scheme employed. On the other hand, this simple approach does not make any use of the additional information contained in video sequences.

The progressive-score-based approach is a simple extension of the previous approach. Instead of classifying frames independently, a simple temporal fusion is applied by accumulating frame scores over time. This can be thought of as classifying every frame as if it were the end of a sequence and taking the final score.

The video-based scheme is much like the progressive-score-based approach but does not return classification results on a frame level, instead classification is done at sequence level. That is, the decision is made based on the final score at the end of the sequence.

Figure 15 demonstrates the development of accumulated scores over a sequence. A reduced training set as explained in Table 12 was used. Even though a few frames are

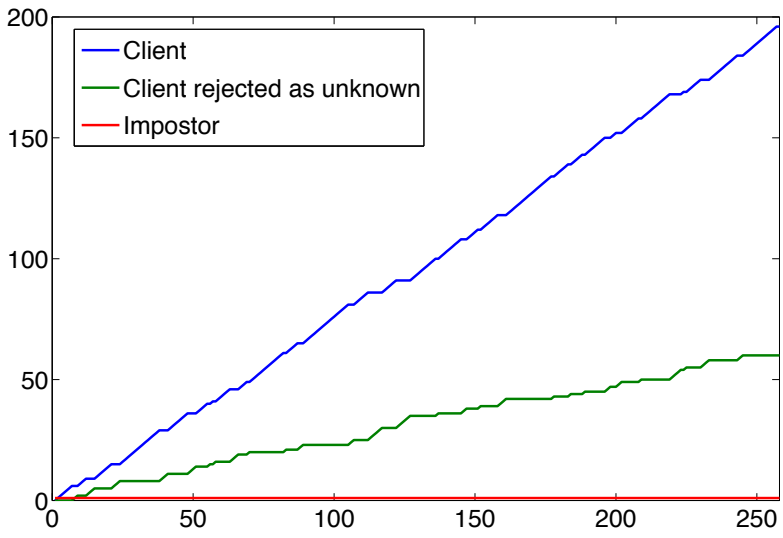


Figure 15: Development of scores for client and impostors

Training data		
Known	5 subjects	2 sessions and ≈ 300 frames per person
Unknown	25 subjects	1 session, 30 frames per subject
Testing data		
Known	5 subjects	3 – 7 sessions per person and 3646 frames overall
Unknown	20 subjects	1 session per person and 3563 frames overall

Table 12: Reduced data set

rejected as unknown or misclassified, the decision is made in favor of the client at the end of the sequence. Training with four sessions would significantly reduce the number of false classifications and rejections.

As can be seen in Table 13 and Table 14, the progressive-score-based approach does already yield very good results. Video-based classification does improve results because decisions are not made on a frame level but on sequence level thus discarding the few early false classifications and rejections of the progressive-score-based approach.

Frame-based classification indicates the system's baseline performance, progressive-score-based classification could be used in systems without fixed decision points, whereas video-based classification may be used in scenarios with fixed decision points, i.e. when there are n_i recorded frames of a person i and the decision is to be made after all frames have been recorded. A system that uses few still images would use frame-based classification, an online system that needs a decision every frame could use the progressive-score-based approach and a system that classifies after a complete sequence has been captured, i.e. after a person has left from in front of the camera, could use

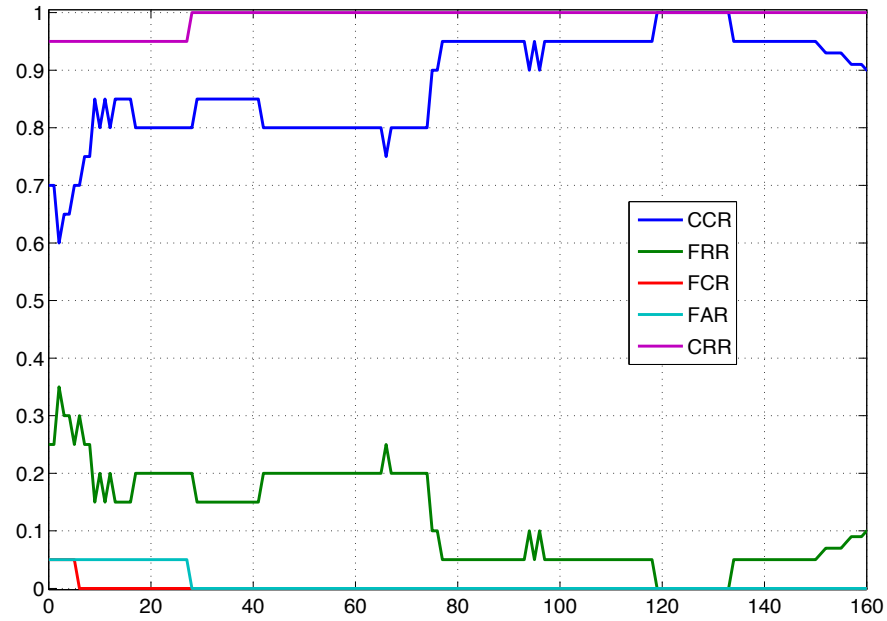


Figure 16: Classification score development by frame number

the video-based approach.

Classification	CCR	FRR	FAR	CRR	FCR
Frame-based	87.2 %	12.5 %	3.7 %	96.3 %	0.3 %
Progressive-score	99.5 %	0.5 %	0.1 %	99.9 %	0 %
Video-based	100 %	0 %	0 %	100 %	0 %

Table 13: SVM classification results

Classification	CCR	FRR	FAR	CRR	FCR
Frame-based	81.6 %	15.5 %	17.8 %	82.2 %	2.9 %
Progressive-score	93.7 %	5.9 %	10.1 %	89.9 %	0.4 %
Video-based	95 %	5 %	15 %	85 %	0 %

Table 14: Nearest-neighbor classification results

Figure 16 illustrates the development of the classification rate on the whole testing set for a given number of frames, it displays, for example, the progressive video-based correct classification rate on all testing sequences when only the first $1, 2, 3, 4, \ldots$ were taken for classification. The data set used for training and testing is explained in Table

3. Figure 16 shows that after ten frames the correct classification rate is except for one occasion above 80% and false acceptance rate is at or below 5%.

4.5.2 Sample augmentation

As previously stated, registration plays an important role. The less precise the registration process is, the more the classification performance decreases. But, it cannot be assumed that registration is perfect. One scheme to overcome this was proposed in [3]. The approach is to generate artificial, augmented samples by deliberately manipulating the alignment. This can be done by shifting the detected eye coordinates in the 4-neighborhood around the original detection of the original image. As that is done on the original image the actual shift of eye positions in the registered image depends on the scale of the original face size because it is enlarged/shrinked to 64x64 pixels in the process of registration. Alternating eye coordinates in the 4-neighborhood yields 24 additional samples for every original frame, thus 25-fold amount of data. In order to reduce the amount of data again these samples were clustered with k-means. For a total of n original frames that were augmented, $k = \frac{n}{25}$ was chosen in order to be able to train with the original amount of available data. The experiment results in Table 15 were obtained with the data set configuration explained in Table 12 that uses two training sessions per known. The results in Table 16 were obtained by using the data set configuration in Table 3 that uses four training sessions per known. Both experiments show improved results.

Sample type	CCR	FRR	FAR	CRR	FCR
Non-augmented	68.2 %	31.8 %	0.4 %	99.6 %	0 %
Augmented and clustered	70.3 %	28.6 %	3.1 %	96.9 %	1.1 %

Table 15: Influence of sample augmentation with two known training sessions

Sample type	CCR	FRR	FAR	CRR	FCR
Non-augmented	87.2 %	12.5 %	3.7 %	96.3 %	0.3 %
Augmented and clustered	92.9 %	6.3 %	12.6 %	87.4 %	0.8 %

Table 16: Influence of sample augmentation with four known training sessions

More samples can be generated by varying the detected eye positions by more than a single pixel. The results in Table 17 compare the classification results obtained by varying the detections by either a single pixel or two pixels in the 4-neighborhood of the original detection on the data set explained in Table 12. Therefore, not 25 times as much data is available in the second row but the 89-fold amount. The clustering parameters were modified accordingly. Classification results improve further but the overall runtime of training and classification triples.

Sample type	CCR	FRR	FAR	CRR	FCR
Non-augmented	68.2 %	31.8 %	0.4 %	99.6 %	0 %
Augmented 4-neighborhood 1 pixel step	70.3 %	28.6 %	3.1 %	96.9 %	1.1 %
Augmented 4-neighborhood 2 pixel steps	74.9 %	22.3 %	13.2 %	86.8 %	2.8 %

Table 17: Improvement of classification results by increasing the neighborhood which is used to generate augmentated samples

Sample augmentation seems to provide a substantial improvement in classification performance. As shown in [3] the augmented training set provides significantly better classification results.

4.6 Overall system performance

This section sums up the overall system performance when all the previously optimizations are used simultaneously. These results were generated on the data set described in Table 3 that was used in most other experiments.

In this experiment sample augmentation, SVM-based classification and progressive score-based classification were used to obtain the final results in Table 18.

	CCR	FRR	FAR	CRR	FCR
Overall system performance	99.6 %	0.2 %	0.2 %	94.2 %	5.8 %

Table 18: Overall system performance using sample augmentation, SVM-based classification, progressive score-based classification

4.7 System runtime performance

This section will present training and testing times of the proposed system. The runtimes were measured on an Intel Quadcore Xeon 3.00GHz with 2 GB RAM. The tests were performed on the full dataset as reported in table 3. The base SVM-based classification algorithm with frame-based classification was used. The system runs in real-time, even on much slower laptop computers.

Classification time is 14ms on average including face detection, registration and classification with 5 known subjects trained.

Table 19: System component's runtime performance

Operation	number of executions	average runtime
Adding known training samples	1	2 ms
Adding unknown training samples	1	269 ms
SVM training	1	21470 ms
Image preprocessing	10541	< 1 ms
SVM classification time	7210	14 ms
Clustering of augmented samples	1	359064 ms

Table 19 details the runtime of system components. Training new clients is the single most time-consuming procedure. Training five subjects with around 600 frames each takes about 21.5 secs. After training, classification can be performed in about 14 ms, excluding image frame capturing but including all preprocessing. The algorithm is sufficiently fast to be run on laptop computers in real-time.

Sample augmentation improves the classification results but also imposes a training overhead for generating and clustering the additional samples. When sample augmentation is used the generation of new samples and clustering add an additional overhead of about 6 minutes to the training time. Once the clustering is performed there is no further performance decrease.

5 Conclusion

This work presents a fully automatic system for open-set face identification. A multi-verification based approach was chosen and nearest-neighbor and support vector machine classifiers were employed. The classification is based on local features extracted using discrete cosine transform. Local features were extracted from image blocks by DCT and concatenated into a feature vector that was additionally normalized.

Experiments were conducted on a data set that was collected for this study. We found that support vector machines outperform the nearest-neighbor classifier which has been shown to be a good classifier in closed-set recognition as reported in [1]. Nearest-neighbor-based classification produced quite good results as well, probably due to the rather large amount of available data.

The performance of the proposed system could be further improved, especially by sample augmentation which increases training time but yields much improved results. Video-based classification and progressive scores also significantly improved the performance. The approach seems to work for larger sets of trained identities but further investigation is necessary to determine the performance with more than ten clients. From the experimental results it has been observed that the system copes well with a time gap of three months between recording of training and testing data.

The approach worked well in the designated scenario. It seems to cope well with illumination changes. Different poses seem to be a greater problem. Future work is necessary to improve the handling of different viewing angles in order to further improve results.

References

[1] H.K. Ekenel, R. Stiefelhagen. Local Appearance-based Face Recognition Using Discrete Cosine Transform. *13th European Signal Processing Conference (EUSIPCO 2005)*. Antalya, Turkey, 2005.

[2] H.K. Ekenel, R. Stiefelhagen. Analysis of Local Appearance-based Face Recognition: Effects of Feature Selection and Feature Normalization. *CVPR Biometrics Workshop*, New York, USA, June 2006.

[3] Johannes Stallkamp. Video-based Face Recognition Using Local Appearance-based Models. *Diploma Thesis*, November 2006

[4] F. Li and H. Wechsler. Open Set Face Recognition Using Transduction *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 27(11), November 2005.

[5] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Science*, pp. 71–86, 1991.

[6] W. Zhao, R. Chellappa and P.J. Phillips. Subspace linear discriminant analysis for face recognition. *UMD*, 1999.

[7] P.N. Belhumeur, J.P. Hespanha and D.J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7): 711 – 720, 1997

[8] S. McKenna, S. Gong and Y. Raja. Face Recognition in Dynamic Scenes. *Proceedings of British Machine Vision Conference*, 1997.

[9] A. Pentland, B. Moghaddam and T. Starner. View-based and modular eigenspaces for face recognition. *CVPR 94*, 1994.

[10] B. Heisele, P. Ho and T. Poggio. Face Recognition with Support Vector Machines: Global versus Component-based Approach. *ICCV 2001*, pp. 688–694, 2001.

[11] E. Hjelmas and B.K. Lee. Face detection: A survey. *Computer Vision and Image Understanding*, 83(3): 236-274, Sept. 2001.

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

[13] A. Akbani, S. Kwek and N. Japkowicz. Applying Support Vector Machines to Imbalanced Datasets. *ECML04*, 2004.

[14] B. Schlkopf. Support vector learning. *DAGM'99*, 1999.

[15] K.R. Muller, S. Mika, G. Ratsch, K. Tsuda, B. Scholkopf An Introduction to Kernel-Based Learning Algorithms. *IEEE TRANSACTIONS ON NEURAL NETWORKS*, 2001.

[16] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. *In Proc. 5th Symp. on Mathematical Statistics and Probability*, volume 1, pages 281-297. University of California Press, 1967.

[17] K. Jonsson, J. Matas, J. Kittler, and Y. Li. Learning support vectors for face verification and recognition. *Proc. IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 208–213, 2000.

[18] M. J. Jones and P. Viola. Fast multi-view face detection. *Technical Report TR2003-96, Mitsubishi Electric Research Laboratories*, Cambridge, MA, USA, July 2003.

[19] D.-H. Kim, J.-Y. Lee, J. Soh, Y.-K. Chung. Real-Time Face Verification Using Multiple Feature Combination and a Support Vector Machine Supervisor. *IEEE Proceedings of the 2003 IEEE International Conference on Acoustics, Speech. & Signal Processing*, 2003

[20] K. Lee, Y. Chung and H. Byun. SVM-based face verification with feature set of small size. *ELECTRONICS LETTERS*, 38(15):787 – 789, 18th July 2002

[21] F. Cardinaux and S. Marcel. Face Verification using MLP and SVM. *XI Journees NeuroSciences et Sciences pour l'Ingenieur (NSI 2002)*, num. 21, 2002

[22] P. J. Phillips. Support vector machines applied to face recognition. *Processing Systems 11*, MIT Press, 1999.

[23] Y. Liu and Y. F. Zheng. One-against-all multi-class SVM classification using reliability measures. *Conference on Neural Networks*, IJCNN '05, Proceedings, 2005.

[24] J.C. Platt. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. *Advances in Large Margin Classiers*, 61–74, MIT Press, 1999.

[25] R. Snelick, U. Uludag, A. Mink, M. Indovina, and A. K. Jain. Large-scale evaluation of multimodal biometric authentication using state-of-the-art systems. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(3):450455, 2005.