Low-Resolution Face Recognition Using Rank Lists

Bachelor Thesis

Matthias Mylaeus

18-915-017

Submitted on

August 21st 2022

Thesis Supervisor

Prof. Dr. Manuel Günther

University of Zurich

Department of Informatics

Bachelor Thesis

Author: Matthias Mylaeus, matthias@mylaeus.ch

Project period: February 21st 2022 - August 21st 2022

Artificial Intelligence and Machine Learning Group
Department of Informatics, University of Zurich

Acknowledgements

I want to thank my supervisor, Prof. Dr. Manuel Günther, for proposing the research topic and supporting me during this thesis. His guidance and feedback helped me achieve this result. I am very grateful to have had the opportunity to dive into the field of face recognition, and with his experience, I gained complete insight. Additionally, I thank Ph.D. candidate Xinyi Zhang for further answering questions.

Finally, I thank the Idiap Research Institute (Anjos et al., 2012, 2017) as well as the maintainers of SCface (Grgic et al., 2011) for providing their resources.

Abstract

The field of automatic face recognition has experienced a significant boost in recent years since the use of artificial neural networks was introduced. Face recognition today poses a critical element for everyday life, supporting various tasks from security, surveillance, and access control all the way to unlocking smartphones. In many different situations, such as changing illumination and faces covered with scarfs or glasses, recognition networks have come to shine and achieve the most accurate results. However, when it comes to recognizing faces from a far distance, they start struggling and leave room for improvement. This thesis discusses the usage of a reference database. Images are compared to it, resulting in a signature. Then, rather than comparing a probe image directly to the gallery, their signatures are compared. The idea of rank lists is set side-by-side with the standardization of those signatures to evaluate whether more accurate results can be achieved. However, for all experiments, results show that the usage of a reference database does not outperform the direct comparison. Further research using more extensive databases and various network models is needed to ensure which approach is more accurate.

Zusammenfassung

Der Bereich der automatischen Gesichtserkennung hat in den letzten Jahren einen grossen Aufschwung erlebt, seit die Verwendung künstlicher neuronaler Netzwerke eingeführt worden ist. Die Gesichtserkennung stellt heute ein Schlüsselelement des täglichen Lebens dar und unterstützt verschiedene Aufgaben von der Sicherheit, Überwachung und Zugangskontrolle bis hin zur Entsperrung von Smartphones. In vielen verschiedenen Situationen, wie z.B. bei unterschiedlichen Beleuchtungsverhältnissen sowie bei bedeckten Gesichtern, haben sich Netzwerke bewährt und erzielen die genauesten Ergebnisse. Wenn es jedoch um die Erkennung von Gesichtern aus grosser Entfernung geht, mangelt es an Genauigkeit. In dieser Arbeit wird die Verwendung einer Referenzdatenbank diskutiert, mit der die Bilder verglichen werden, um eine Signatur zu erstellen. Anstatt ein Probebild direkt mit der Galerie zu vergleichen, werden jene Signaturen verglichen. Die Idee der Ranglisten wird mit der Standardisierung dieser Signaturen in Kontrast gestellt, um zu bewerten, ob genauere Ergebnisse erzielt werden können. Bei allen Experimenten zeigt sich jedoch, dass die Verwendung einer Referenzdatenbank den direkten Vergleich nicht übertrifft. Weitere Untersuchungen mit grösseren Datenbanken und verschiedenen Netzwerkmodellen sind erforderlich, um sicher zu sein, welches der genauere Ansatz ist.

Contents

Chapter 1

Introduction

Facial recognition today is so firmly anchored in everyday life that it is barely noticed anymore. People simply unlock their devices such as smartphones and laptops simply by looking at them, and pictures of the same person are naturally grouped in galleries. Furthermore, airports rely on face recognition for border control. As one of the most significant usages, surveillance cameras are installed for monitoring and providing security, identifying criminals or missing people (Masi et al., 2018). For its many applications, it is therefore of great importance that face recognition systems are robust, accurate, and efficient.

Given its many advantages, especially for law enforcement, research on facial recognition is pushed heavily (Zhao et al., 2003). And rightly so, since even though face recognition systems have matured to a particular level, researchers agree that their performance and success are still heavily limited by certain real-world situations. In *controlled* environments, recognition systems outperform human abilities (O'Toole et al., 2007). That is, using images with perfect illumination with the subject's head turned to the front and with no facial expressions or background noise. In everyday life, however, this situation is not very likely to exist. Surveillance cameras usually record people from an elevated position. Hence people's heads are rarely turned to the camera. People like to wear sunglasses and hats in summer and scarves in winter, covering crucial parts of their faces. Furthermore, a person's face is not printed in stone but changes with age. People's faces express emotions and therefore are rarely neutral. Finally, depending on the time of day and artificial lighting, illumination greatly varies, making it challenging to recognize facial characteristics.

Consequently, it is not surprising that face recognition systems perform worse in *uncontrolled* environments. For a significant part, the human visual system can easily separate identities no matter how the factors mentioned above change. Because of the many faces people are confronted with throughout their life, their visual system is extensively trained (Müller et al., 2013). Meanwhile, automatic face recognition systems have progressed so far, that differences in poses as well as occlusions (i.e., scarfs and sunglasses) practically do not impact the results (de Freitas Pereira et al., 2022). However, achieving similar efficiency using automatic face recognition is a challenge yet to be tackled concerning other areas such as when working with low-resolution footage.

Over the years, there have been many different approaches. Traditional ones include algorithms such as Linear Discrimination Analysis, Principal Component Analysis and Local Gabor Binary Pattern Histogram Sequence. However, the most accurate results have been achieved since the introduction of machine learning. Artificial neural networks are trained with thousands of images to extract facial characteristics (i.e., features) to gain the ability to recognize complex abstractions of faces (Wang and Deng, 2021). Since the breakthrough in 2012 described by Krizhevsky et al. (2012) in deep learning, the field of face recognition has been dominated by neural networks.

The research on image databases for training such networks is a whole branch itself. Much previous work is published using proprietary databases and is therefore difficult to verify. Researchers such as Grgic and Delac[1] work towards publicly available ones, which additionally define evaluation protocols that declare which images are used for *training* or *evaluation*. Experiments made using such open-source databases are re-runnable and verifiable, which poses a significant advantage for future work.

Facial recognition can be divided into two parts: identification and verification. For both applications, different databases have to be considered. Identification describes the process where the identity of a recorded subject is deciphered by comparing the image to a gallery. This is the typical application performed by surveillance cameras which, for instance, record and identify people who want to access a restricted area. Verification, on the other hand, deals with validating whether people indeed are who they claim to be. For instance, Uber drivers are asked to take a picture of themselves after uploading their driver's license for validation.[2]

For this thesis, the SCface database (Grgic et al., 2011) designed mostly for identification purposes is used, which provides low-resolution images taken with state-of-the-art surveillance cameras. The present work aims to set side-by-side the following two approaches: the direct comparison of a low-resolution image with a high-resolution frontal mugshot; and an indirect comparison with the help of a reference database, also called a cohort. For this, the rank list method is applied as explained by Schroff et al. (2011) and Müller et al. (2013). A rank list derived from an image's signature is compared to other rank lists. Furthermore, another approach is deduced, which also uses the cohort however does not convert the signatures into rank lists but standardizes them before comparison.

Chapter 2 provides an extensive literature overview on the history of face recognition, its composing stages, deep learning, as well as on the rank list approach, including various comparison methods. Chapter 3 details the background relevant for this thesis and consists of the SCface database, the network model for feature extraction, and the software used for computational support. This is followed by a description of the approach in chapter 4 and the different methods used for achieving the results in chapter 5. Finally, the results are discussed in chapter 6, and the paper is concluded in chapter 7. All implementation details, as well as additional results and plots, are recorded in the appendix A.

[1]`https://www.face-rec.org/databases/`, retrieved July 14, 2022.

[2]`https://help.uber.com/driving-and-delivering/article/identity-verification-checks?nodeId=75fb55dc-da08-41bb-b1cb-3229f2839956`, retrieved July 14, 2022.

Chapter 2

Related Work

This chapter highlights previous work on face recognition and deep learning. The origins and development period of face recognition and the common recognition stages necessary for matching identities are mentioned. The idea of comparing images with the help of rank lists, which are generated using a reference database, is described. This method is mainly based on previous research by Müller (2010), Schroff et al. (2011) and Müller et al. (2013) which is summarized additionally.

2.1 An Overview of Face Recognition

The origins of automated face recognition date back to the 1960s, when the pioneers Woody Bledsoe, Helen Chan Wolf, and Charles Bisson first worked on using a computer to recognize human faces. The problem was to find a record in a large dataset of images that best matched a given photograph. Due to the investor's constraints on publicity, little work was published (Shaha et al., 2008). Still, Bledsoe et al. proved that facial recognition could pose a viable biometric. By manually marking facial landmarks (e.g., eyes and mouth), they would calculate the distances between the features, which were then compared between images to find a matching identity (de Leeuw and Bergstra, 2007). However, the development stage of facial recognition for businesses did not start until the late 1980s, when the discovery of a system today known as Eigenfaces was made, enabling the formation of a set of basic features from facial images automatically (Sirovich and Kirby, 1987).

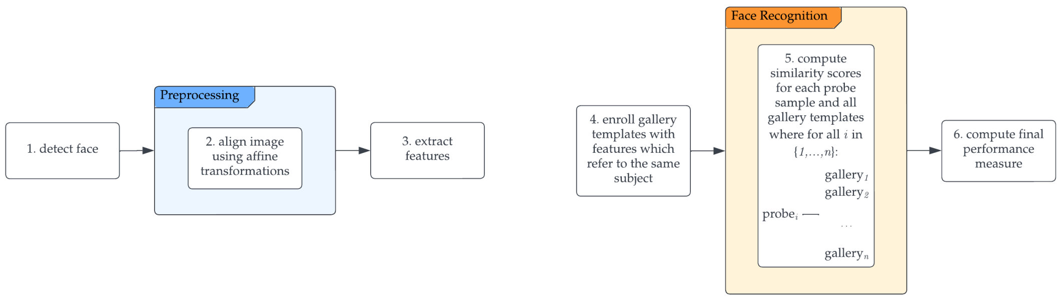
In 2007, O'Toole et al. (2007) showed that in *controlled* environments, automatic face recognition systems could outperform human abilities and correctly match more identities. Researchers then moved on to *uncontrolled* environments, where many factors strongly influence face recognition algorithms. The most well-known are facial pose, illumination, camera type, distance to the camera, position within the camera view, accessories (e.g., hats or scarfs), facial expression, and aging (Müller et al., 2013).

Ever since many face recognition algorithms have been published and matured into a stage where they are used in various daily activities (e.g., unlocking devices and identification at border control). However, most published works were built upon image databases not available to the public; hence only report results for which it is unclear which parts of the databases are used (Günther et al., 2017). In addition, the complexity of research challenges and the lack of software including all requirements lead to many non-reproducible papers. To counteract this problem, researchers try to avoid using proprietary resources and work on solutions to ease and motivate the use of open-source software and data (Günther et al., 2012b; Anjos et al., 2012; Günther et al., 2017). Günther et al. (2016), for instance, published their study (which they further extended in 2017) solely based on such open-source material. By using resources available to the public, the experiments are reproducible, and performance comparisons between various algorithms are verifiable.

Today, the field of face recognition is dominated by deep learning. Artificial neural networks (i.e., a network of artificial neurons constituting a human brain) are trained with various databases and achieve accurate results, revolutionizing the traditional approach (Wang and Deng, 2021).

2.2 Recognition Stages

Günther et al. (2016, 2017) describe the stages into which the face recognition task can be divided. Traditional preprocessing steps include algorithms for applying a photometric enhancement which reduces the effects of illumination conditions. However, thanks to deep learning, preprocessing is a lot simpler. An overview of the complete face recognition pipeline, which is applied nowadays, is given in figure 2.1.


1. detect face
Preprocessing
2. align image using affine transformations
3. extract features
4. enroll gallery templates with features which refer to the same subject
Face Recognition
5. compute similarity scores for each probe sample and all gallery templates where for all i in $\{1,\ldots,n\}$:
probe$_i$
gallery$_1$
gallery$_2$
…
gallery$_n$
6. compute final performance measure

(a) Preprocessing and feature extraction steps for every image of the database.

(b) Scores calculation between probe and gallery samples for identification.

Figure 2.1: RECOGNITION STAGES. This figure including subfigures (a) and (b) shows the order of execution when running the face recognition task.

2.2.1 Preprocessing

Traditional algorithms for preprocessing include Histogram Equalization (HE) (Ramírez-Gutiérrez et al., 2010), Self Quotient (SQ) (Wang et al., 2004), Tan & Trigg's (T&T) (Tan and Triggs, 2010), and Local Binary Patterns (LBP) (Heusch et al., 2006), as mentioned by Günther et al. (2017). These, however, are not necessary when using artificial neural networks. After detecting the face, all images are aligned using affine transformations. Each pixel in the original image is transformed, which results in a new image that is geometrically normalized and already cut to a defined resolution. The preprocessed image is then simply fed to a trained neural network for extracting facial features (Guo and Zhang, 2019).

2.2.2 Face Recognition

Once the facial features are extracted from all the images in the database, gallery templates are enrolled for features that refer to the same subject. These templates are then used in the actual face recognition step. The features of one probe image are compared to the templates of each gallery subject to compute a similarity score. This step is the most important for this thesis as different methods for calculating these scores will be discussed. Günther et al. (2017) mention various traditional algorithms, listed below, in charge of this step before the usage of neural networks. Depending on the algorithm, the type of features may differ.

- **Linear Discrimination Analysis (LDA):** features are projected into a subspace, where they are compared by a distance measure to maximize class separability (Zhao et al., 1998).
- **LDA-Infrared (LDA-IR):** additionally to LDA, two color layers of the image provide more information (Lui et al., 2012).
- **Principal Component Analysis (PCA):** the projection is the same as the one for LDA. However, PCA finds the directions of maximal variance (Turk and Pentland, 1991).
- **Local Region PCA (LRPCA):** this algorithm computes PCAs for several local regions (e.g., eyes, nose, mouth) (Phillips et al., 2011).
- **Local Gabor Binary Pattern Histogram Sequence (LGBPHS):** this algorithm uses the LBP preprocessing algorithm. LBP codes are combined into several histograms used for comparison (Zhang et al., 2005).
- **Gabor Grid Graphs:** so-called Gabor jets described as local texture features are extracted and used to find the similarity between images (Günther et al., 2012a).
- **Inter-Session Variability (ISV):** with the help of Gaussian mixture models, images can be compared and evaluated for similarity (Wallace et al., 2011).

2.2.3 Evaluation: Identification & Verification

Finally, after computing all similarity scores, the final performance measure is computed. Identification Performance (IP) is reported as a Recognition Rate (RR). The RR is the total number of correctly identified probe images (i.e., genuines) denoted as gen divided by the total number of all probe images N_p and therefore shows the percentage of success:

$$\text{RR} = \frac{gen}{N_p} \tag{2.1}$$

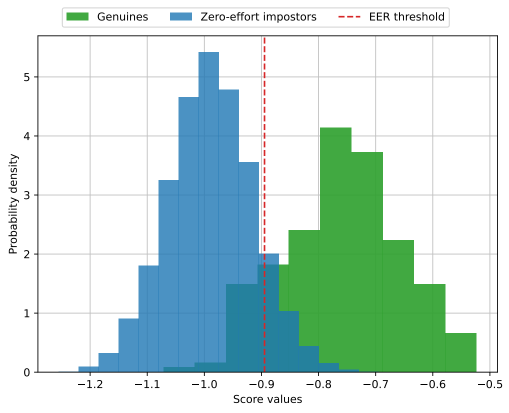
Verification Performance (VP) can be reported in several ways. Commonly used are the False Match Rate (FMR) and the False Non-Match Rate (FNMR), similar to False Acceptance Rate (FAR)/False Positive Rate (FPR) and False Rejection Rate (FRR)/False Negative Rate (FNR), respectively. Both FMR and FNMR are calculated using a threshold θ. FMR checks the total number of falsely matched probe-gallery pairs (i.e., impostors) denoted as imp with a score above θ, whereas FNMR checks the total number of genuines with a score below θ. Dividing by the total number of impostors/genuines respectively will result in the final rate:

$$\text{FMR}(\theta) = \frac{imp > \theta}{imp} \qquad \text{FNMR}(\theta) = \frac{gen < \theta}{gen} \tag{2.2}$$

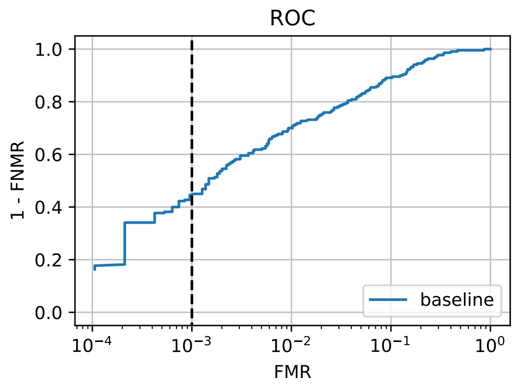
The two rates are commonly plotted in two ways:

1. As a histogram of genuines, impostors, and an Equal Error Rate (EER). The EER indicates at what threshold θ the FMR equals the FNMR.
2. Using the Receiver Operating Characteristics (ROC) which plots the True Match Rate (TMR) calculated by $1-$FNMR over the FMR.

Figure 2.2 shows an example of each evaluation metric.



(a) Genuines, Impostors and EER



(b) ROC

Figure 2.2: VERIFICATION PERFORMANCE. This figure including subfigures (a) and (b) shows an example of each evaluation metric for VP.

2.3 Deep Learning

Deep learning has become very popular in face recognition in recent years. The main difference when using deep learning is how facial features are extracted from images. As mentioned, traditionally, algorithms such as HE or LBP would work with hand-labeled annotations of facial landmarks (i.e., eyes and mouth) to successfully extract the features. Depending on the feature extraction, various algorithms such as PCA or LGBPHS would then be used to compare those features. However, when applying deep learning methods, the preprocessed images are fed to a neural network. It takes the values of each pixel of the images and outputs a vector representing the subject's identity (i.e., the subject's features). Features are then compared with the help of distance computations such as the cosine distance (2.4) (Wartmann, 2021).

But what exactly is a neural network? Guresen and Kayakutlu (2011) describe an artificial neural network as a mathematical model inspired by the structure of the human brain. Namely, neurons that are connected through weighted links and process information to overcome complex computational tasks. Activation functions determine when neurons are meant to fire and when not (Guo and Zhang, 2019). Commonly, networks consist of an input, a hidden data processing, and an output layer. However, they are not restricted to a single data processing layer. Several hidden layers which take the output of the previous one as new input can be connected. These networks are then called deep neural networks (Wang and Deng, 2021). The network most widely used for face recognition tasks is the so-called Convolutional Neural Network (CNN) (Yi et al., 2014). CNNs are made up of three hidden layers: convolutional layers, which take the input and extract essential features; fully-connected layers, which support linking together neurons & layers; and pooling layers used for down-sampling (Guo and Zhang, 2019).

In addition to its architecture, networks require training data. The more training data is available, the more accurate the network becomes, similar to how humans automatically train their visual system when being confronted with many faces. Parkhi et al. (2019) propose a procedure to create a reasonably large face dataset. First, a list of names of candidate identities is required to obtain faces. By focusing on celebrities and public figures, a large number of distinct images is guaranteed. Next, each name is queried in online search engines (e.g., Google) to obtain enough pictures. Third, erroneous faces are filtered from the images by using the top 50 images of one subject as positive and the top 50 of all the other subjects as negative training samples. Finally, before further purifying the data using human annotations, duplicate images and near duplicates from the same subject are removed. Using this method, a dataset with roughly 950'000 good images can be generated (Parkhi et al., 2019).

Next to a network's architecture and training data, the usage of loss functions is essential for accurate results. Loss functions serve the purpose of maximizing intra-class similarity and inter-class variance. Intra-class similarity describes how close a subject's features are to one another, whereas inter-class variance describes the difference between the features of two subjects. Therefore, the values in one feature vector are ideally close together while being very different from those of another (Schmidli, 2021).

Creating different neural networks takes various architectures, loss functions, and training data. Naturally, the more accurate a network has to be, the more complex and extensive it becomes. AlexNet, published in 2012, was the first network to achieve state-of-the-art results and marks the breakthrough of deep learning (Krizhevsky et al., 2012). Since then, many new architectures and loss functions to create networks with improved performance have been published. Widely used loss functions include softmax loss (Cao et al., 2018), triplet loss (Schroff et al., 2015), and center loss (Wen et al., 2016) which are all Euclidean-based (i.e., they separate features using the Euclidean distance). Recent trends show the introduction of cosine-margin-based functions, which try to separate features using the larger cosine distance instead. This should further maximize face class separability. Well-known functions include large-margin loss (or L-Softmax) (Liu et al., 2016), SphereFace (Liu et al., 2017), CosFace (Wang et al., 2018), and ArcFace (Deng et al., 2019). For the thesis at hand, solely ArcFace is of importance and mentioned in section 3.2.

2.4 Rank Lists

Commonly when performing the face recognition task, the probe images are each compared to every gallery image directly. There are many methods to determine similarity; this thesis focuses on the cosine similarity measure to compute the scores for direct comparison. It takes two one-dimensional vectors u and v (i.e., the extracted features of one probe and one gallery image), where higher values stand for higher similarity. The cosine distance $d_{cos}(u,v)$ can be derived from the similarity as follows:

$$S_{cos}(u,v) = \frac{u \cdot v}{||u|| ||v||} \tag{2.3}$$

$$d_{cos}(u,v) = 1 - S_{cos}(u,v) \tag{2.4}$$

However, while yielding good results in most cases, a direct comparison of probe and gallery images is not always the best approach. As mentioned in section 2.1, performance is influenced by many factors in *uncontrolled* environments.

2.4.1 Idea

Before the usage of networks, Schroff et al. (2011) found that when comparing images of subjects with different poses and expressions, better results can be reached by using so-called doppelgänger lists (i.e., rank lists). These lists are generated with the help of a reference database (also known as a cohort), by first creating a signature for each image which is then converted into ranks. Based on the following two ideas, rather than comparing the features directly, the similarity of those rank lists can be computed to find the best match.

1. People who are similar in one situation (e.g., with their head turned to the side) will also be similar in another (e.g., with their head turned to the front).

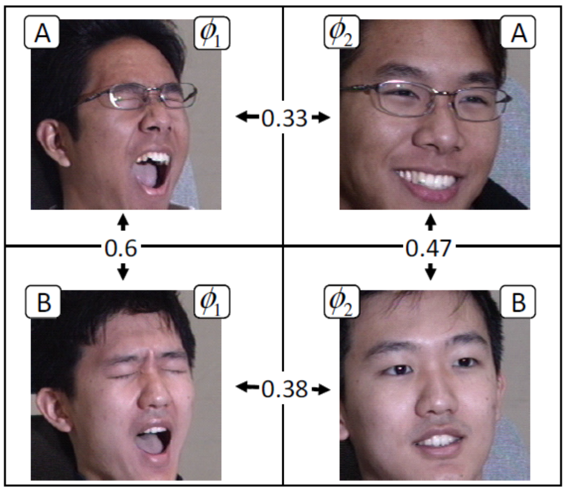


Figure 2.3: RECOGNITION ACROSS EXPRESSION. This figure taken from Schroff et al. (2011) illustrates the impact of facial expressions before the introduction of deep learning. Comparing person A in the top and person B in the bottom row for the two situations ϕ_1 and ϕ_2 results in a higher similarity for expressions alike rather than for the same person.

2. People can be described in relation to other people (e.g., a person A looks similar to a person B but not to a person C)

The cohort for this method consists of images with subjects different from the ones used in probe and gallery images. Schroff et al. (2011) explain that when using distance measures like described in formula (2.3), the cohort subject with the smallest distance to the probe/gallery subject will become a look-alike (i.e., a doppelgänger). All the images in the cohort can be sorted following this procedure, resulting in a doppelgänger list.

Schroff et al. (2011) show that this method works especially well for subjects with different poses and expressions because the subjects considered the most similar from the cohort are likely to have similar poses and expressions. Therefore, one could say that the comparison considers more information and can achieve better results than when comparing only features directly with each other. This is further shown in figure 2.3.

To visualize the process of the rank list method, figure 2.4 shows the separate steps to calculate similarity scores between probe and gallery images. The first step is to compute $S_{cos}(u, v)$ for the features of all the probe and gallery images separately, each with those of the cohort images which show the same variation. Therefore, after this step, every probe and gallery image is assigned a one-dimensional vector filled with similarity scores (i.e., its signature) each referring to one subject in the cohort. Next, these vectors are converted into rank lists. This is done by giving the first rank 0 to the highest entry (i.e., the most similar cohort image), the next rank 1 to the next highest, and so on. The last step is to compute the final similarity scores between rank lists (i.e., all lists of the probe images, each with the ones of the gallery images). Figure 2.5 visualizes how the cohort is applied.

Today, neural networks have progressed so far, that differences in poses as well as occlusions (i.e., scarfs and sunglasses) practically do not impact the results anymore (de Freitas Pereira et al., 2022). Especially the network model ArcFace-100 shows solid performance. However, since a lot of the state-of-the-art networks still show inaccurate results when using low-resolution images, the rank list method is important for this thesis.

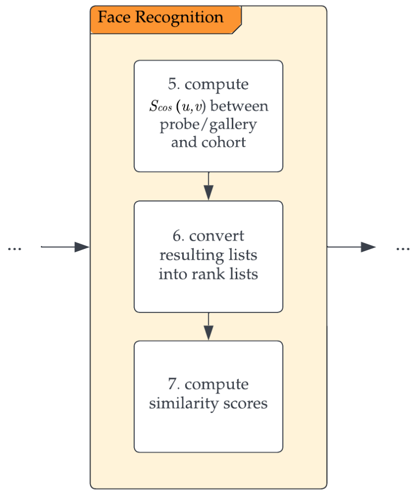

Face Recognition
5. compute $S_{cos}(u,v)$ between probe/gallery and cohort
6. convert resulting lists into rank lists
7. compute similarity scores
...
...

Figure 2.4: RANK LIST STAGES. This figure shows the order of execution when running the rank list method according to Schroff et al. (2011) extending figure 2.1.

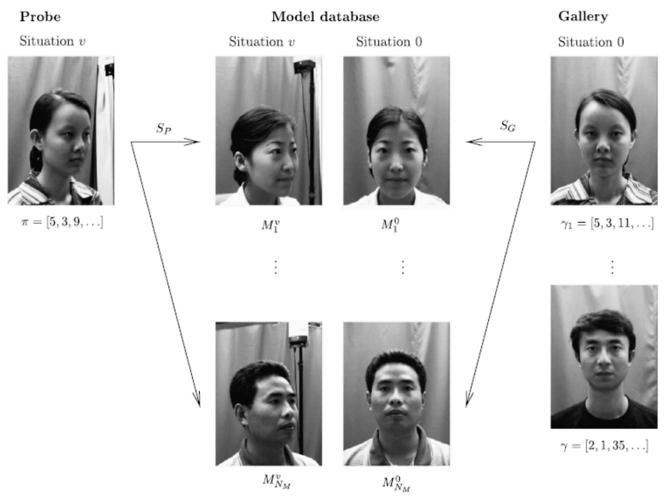


Figure 2.5: SIGNATURE GENERATION. This figure taken from Müller et al. (2013) shows how the cohort is used to generate an image's signature, which is then converted to a rank list.

2.4.2 Comparison Methods

Müller 2010

The first usage of rank lists is described by Müller et al. (2007) and Müller (2010). Müller (2010) introduces the following terminology: let G stand for all gallery, P for all probe, and C for all cohort images, where N denotes the number of subjects in the cohort. For the first step as described above, each probe p (i.e., its features) and each gallery image g separately is assigned a list of similarity scores. The scores are computed by referring to each cohort image c with the same variation (e.g., taken at equal distance) using formula (2.3):

$$\forall m \in \{1, \dots, N\} : S_{cos}(p, c_m) \quad \text{and} \quad \forall m \in \{1, \dots, N\} : S_{cos}(g, c_m) \tag{2.5}$$

The resulting lists are converted into rank lists π for probe and γ for gallery images. For comparing rank lists, Müller (2010) describes three criteria necessary for efficiently computing the similarity:

1. Identical rank lists must result in the highest similarity (i.e., the output of $S \in [0, 1]$ with $S = 1$ indicating maximal similarity).
2. The more equal indices result in the same rank, the higher the similarity should be.
3. Lower ranks (i.e., showing high similarity) should be weighted more than higher ranks.

Müller (2010) proposes the following function, which fulfills all criteria and computes the similarity between two rank lists, where F denotes a normalization factor:

$$\begin{aligned} S_{2010mueller}(\pi, \gamma) &= \frac{1}{F} \cdot \sum_{m=0}^{N-1} \frac{1}{\sqrt{\pi(m) + \gamma(m) + 1}} \\ F &= \sum_{i=0}^{N-1} \frac{1}{\sqrt{2i + 1}} \end{aligned} \tag{2.6}$$

The similarity scores are then further used to evaluate IP and VP.

Schroff

Schroff et al. (2011) found that the similarity value drops significantly at some point and that the differences become small after this point. Therefore, instead of applying weights according to the rank, they include a factor k. Only ranks up to this factor are considered, where similarities are most significant. They introduce the following similarity function. Note that the factor k is not increased by one as in the original function since the lowest rank in the lists π and γ is zero.

$$\begin{aligned} S_{schroff}(\pi, \gamma) &= \sum_{m=0}^{N-1} \left[k - \pi(m)\right]_+ \cdot \left[k - \gamma(m)\right]_+ \\ [x]_+ &= \begin{cases} x \text{ if } x > 0, \\ 0 \text{ otherwise} \end{cases} \end{aligned} \tag{2.7}$$

Note that due to $[x]_+$, the results using values where $k > N$ are equal to the those achieved with $k = N$. Similarly, when using values where $k < 0$ the performance remains equal to that achieved with $k = 0$.

Müller 2013

Two years later Müller et al. (2013) published their article including a new similarity function, which again fulfills all mentioned criteria and improves $S_{2010mueller}$ (2.6):

$$\begin{aligned} S_{2013mueller}(\pi,\gamma) &= \frac{1}{F} \cdot \sum_{m=0}^{N-1} \lambda^{\pi(m)+\gamma(m)} \\ F &= \sum_{i=0}^{N-1} \lambda^{2i} \end{aligned} \tag{2.8}$$

Instead of the decaying function $f_1(x)$ used for $S_{2010mueller}$ (2.6), $f_2(x)$ is used noted below. It guarantees a faster decay and therefore yields better results. Additionally, it "[...] allows for a natural interpretation and implementation as a neural network"(Müller et al., 2013).

$$f_1(x) = \frac{1}{\sqrt{x+1}}, \qquad f_2(x) = \lambda^x \text{ with } \lambda \in [0.9, 1) \tag{2.9}$$

Wartmann

When working on frontal-to-profile face recognition with rank lists, Wartmann (2021) however, found that weighing low ranks more than high ranks does not necessarily result in the best scores. Instead, Wartmann weighted low ranks as well as high ranks more than ranks in between achieving better results, and redefined the criteria a similarity function should fulfill as follows:

1. Rank differences of low as well as high ranks should be weighted more.
2. The more minor absolute differences between ranks are, the more similar the respective rank lists should be.
3. High absolute differences should have the opposite effect (i.e., the rank lists should be different).

In regard to these criteria Wartmann (2021) defined a new parametric similarity function, where the parameters $\alpha, \beta \geq 1$:

$$\begin{aligned} S_{wartmann}(\pi,\gamma) &= -\sum_{m=0}^{N-1} f_\alpha\big(\pi(m),\gamma(m)\big) \cdot f_\beta\big(\pi(m),\gamma(m)\big) \\ f_\alpha(x,y) &= \left(\frac{|x-y|}{N}\right)^\alpha \\ f_\beta(x,y) &= \left(\left|\frac{2x}{N}-1\right|^\beta + \left|\frac{2y}{N}-1\right|^\beta\right) \end{aligned} \tag{2.10}$$

Here, f_β is used for weighting high and low ranks more heavily than intermediate ones (criterion 1). β regulates how extreme (i.e., how high or how low) the rank must be for being weighted more heavily; thus, large β values result in a smaller range of heavily weighted ranks. f_α can be understood as weighting function for f_β with values between 0 and 1. Additionally, the exponent α guarantees the increased weighting of large differences between ranks compared to small differences (criteria 2 & 3). Note that the sum is negated such that higher scores stand for higher similarity.

Correlation Functions

Even though rank lists were first used and compared by Müller et al. (2007) and Müller (2010), the mentioned statistical methods are not the first for comparing permutations. Traditional correlation functions can also be applied to rank lists, such as the Spearman correlation (Spearman, 1987):

$$S_{spearman}(\pi,\gamma) = 1 - \left(\frac{6}{N(N^2-1)} \cdot \sum_{m=0}^{N-1} \big|\pi(m) - \gamma(m)\big|^2\right) \tag{2.11}$$

The function counts the differences in the positions of the same ranks. In fact, it is very similar to $S_{schroff}$ (2.7), which sums the products of both ranks. When expanding the binomial of $S_{spearman}$ (2.11), it becomes clear that both functions behave similarly, as the first part of $S_{spearman}$ (2.11) can be seen as a normalization factor. Even more so, when $k = N$ is chosen; then the two functions become monotonous.

Another correlation is described by Kendall's tau (Kendall, 1938), for which a weighted version was later implemented (Vigna, 2015).

$$\begin{gathered} S_{kendall}(\pi,\gamma) = \frac{2}{N(N-1)} \cdot \sum_{m_1=1}^{N-1} \sum_{m_2=0}^{m_1-1} sgn\big(\pi(m_1) - \pi(m_2)\big) \cdot sgn\big(\gamma(m_1) - \gamma(m_2)\big) \\ sgn = \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0 \end{cases} \end{gathered} \tag{2.12}$$

Kendall's tau counts how many pairs of ranks switched direction between the lists. The weighted version of Kendall's tau adds weights according to the rank using the following function $f_3(x)$. Adding weights guarantees the third criterion by Müller (2010).

$$f_3(x) = \frac{1}{x+1} \tag{2.13}$$

Therefore, the correlation function is defined identical to $S_{kendall}$ (2.12) with the difference that each rank is weighted first:

$$\begin{aligned} S_{w.kendall}(\pi,\gamma) = \frac{2}{N(N-1)} \cdot \sum_{m_1=1}^{N-1} \sum_{m_2=0}^{m_1-1} & sgn\Big(f_3\big(\pi(m_1)\big) - f_3\big(\pi(m_2)\big)\Big) \\ & \cdot sgn\Big(f_3\big(\gamma(m_1)\big) - f_3\big(\gamma(m_2)\big)\Big) \end{aligned} \tag{2.14}$$

Chapter 3

Background

This chapter describes the setup and software used for running the experiments: the SCface database, which provides images taken in different situations and defined protocols; the library InsightFace, which enables feature extraction; the open-source software Bob used for pipeline setup and evaluation; and the open-source software SciPy which provides correlation functions and distance measures.

3.1 SCface Database

There are many different face databases available today. On their website, Grgic and Delac[1] list a total of 101 databases often used by researchers. Many, however, include images taken in a *controlled* environment and are of high resolution. Pictures are taken not using commercially available surveillance equipment and usually using the same camera for all of them. Grgic et al. (2011) claim that the images "[...] lack the real-world settings part." and are therefore far from real-world conditions (i.e., an *uncontrolled* environment). This leads to the issue that there are few studies "[...] on face recognition in such naturalistic conditions, resulting in very high recognition rates suggesting that face recognition is almost a solved problem." (Grgic et al., 2011).

The SCface database should fill the gap as it mimics an *uncontrolled* environment. It consists of 4'160 images from 130 subjects taken in uncontrolled lighting with five different surveillance video cameras. Additionally, a high-resolution facial mug shot for each subject is included (cropped to 1'600 $\times$ 1'200 pixels with a face height of roughly 320 pixels). Each subject is recorded at three different distances resulting in various quality and resolution. Protocols indicate which images of the gallery and the probe are compared (i.e., images of a particular variation such as the distance). Since the gallery of the SCface database only consists of the mugshots, the images from it are the same for each protocol. However, the protocols help filter the probe images according to their distance:

- **Distance 1** [protocol *far*]**:**
  images taken with a distance of 4.2 meters cropped to 100 $\times$ 75 pixels, where the face only has about 20 pixels of height (de Freitas Pereira et al., 2022).

- **Distance 2** [protocol *medium*]**:**
  images taken with a distance of 2.6 meters cropped to 144 $\times$ 108 pixels, with a face height of roughly 30 pixels.

[1] `https://www.face-rec.org/databases/`, retrieved July 14, 2022.

- **Distance 3** [protocol *close*]**:**
  images taken with a distance of 1.0 meter cropped to 224 $\times$ 168 pixels, where the height of the face amounts to approximately 45 pixels.

Since the cameras are placed slightly above the subject's head (i.e., installed at the height of 2.25 meters), the pose is typical for a commercial surveillance system. The database also includes infrared photos and pictures with different poses per subject (from -90 to $+90$ degrees); however, these are not considered in this thesis and, therefore, not significant. Figure 3.1 shows the images of one subject from the database relevant to this thesis.
Subjects include 114 males and 16 females, all Caucasian, most of them between the ages of 21 and 23 (see figure 3.2). Textual files contain information such as whether the subject has a beard, a mustache, or is wearing glasses (Grgic et al., 2011).

The SCface database is of interest for this thesis because it provides low-resolution surveillance footage necessary to compare methods that might yield more accurate results than many state-of-the-art face recognition algorithms.

3.2 Face Analysis Library InsightFace

As mentioned in section 2.3, using artificial neural networks is the method of choice for face recognition. CNNs are trained on various databases to successfully and accurately map the face images into features. Teaching the CNN determines its performance and is therefore very important. As mentioned earlier, one of the main challenges is "[...] the design of appropriate loss functions that enhance discriminative power" (Deng et al., 2019). Deng et al. (2019) introduce an additive angular margin loss called ArcFace, which:

- optimizes the distance margin, which guarantees small intra-class and large inter-class distances,
- consistently outperforms the state-of-the-art,
- is easy to implement, as it does not have to be combined with other loss functions to show stable performance
- and runs efficiently with negligible computational overhead; therefore is applicable with a large number of identities.

For the experiments, a model which has been trained on the face analysis library InsightFace is used. The library includes, next to other state-of-the-art algorithms of face recognition, face detection, and face alignment, several models of the ArcFace recognition approach. As it proves such outstanding performance (de Freitas Pereira et al., 2022), the model ArcFace-InsightFace-IResNet100 trained on the MS1MV2 dataset[2] is applied for feature extraction.

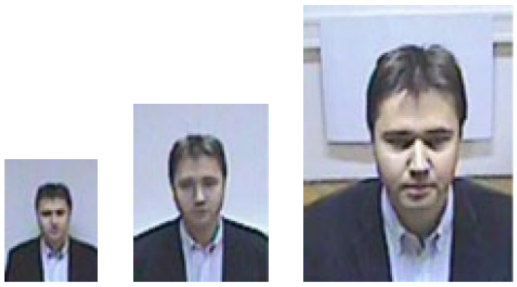
3.3 Software Bob

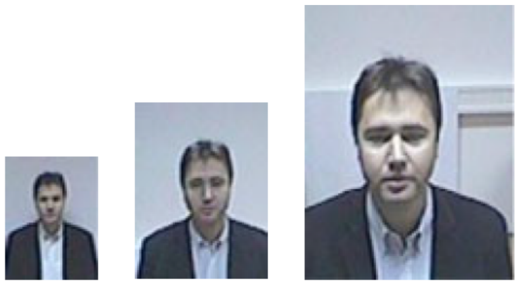
Section 2.1 briefly mentions how the works of Günther et al. (2016) and Günther et al. (2017) influence the reproducibility of experiments. Anjos et al. (2017) define four criteria a paper must provide to be considered reproducible.

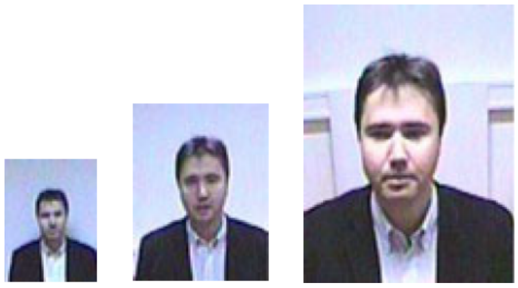
[2]https://github.com/deepinsight/insightface/tree/master/model_zoo, retrieved July 14, 2022.

(a) Mugshot

(b) Surveillance Camera 1


(c) Surveillance Camera 2


(d) Surveillance Camera 3


(e) Surveillance Camera 4

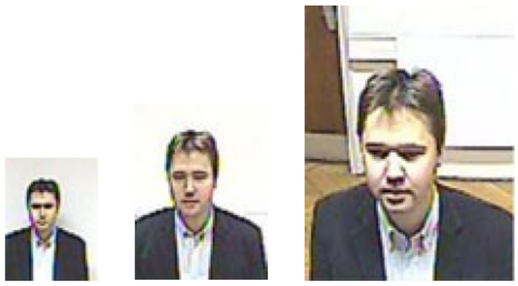

(f) Surveillance Camera 5

Figure 3.1: SCFACE IMAGE SET. This figure taken from Grgic et al. (2011) including subfigures (a) - (f) shows an image set of one subject from the database (only including the images of importance for this thesis).

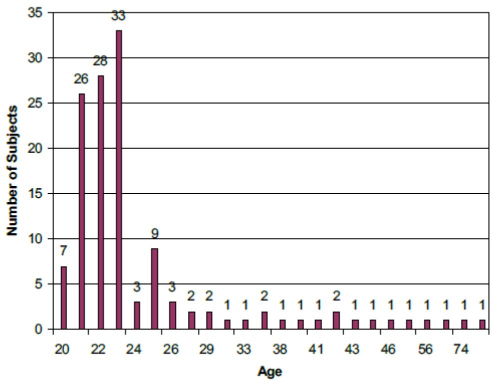


Figure 3.2: SCFACE AGE DISTRIBUTION. This figure taken from Grgic et al. (2011) shows the age distribution of the subjects recorded in the SCface database.

1. All experiments declared should be **repeatable** yielding the same results choosing the same parameters.

2. The data and code used should be **open-source**, such that it can be shared and anybody can re-run the experiments.

3. The infrastructure should be easy to understand and **open for extension** to enable implementation of new approaches.

4. The system should run reliably and **prove stability**.

The software Bob is an open-source, all-in-one, transparent signal processing and machine learning toolbox designed per these criteria (Anjos et al., 2012). It provides a researcher-friendly Python environment reducing development time and guarantees efficient processing with its fast C++ implementations. Several implementations for preprocessors, feature extractors, databases, recognition algorithms, and evaluation metrics are available. For databases, Bob includes the available evaluation protocols which ensure reproducibility. They define the images for *training* as well as the ones for *evaluating* an algorithm. Furthermore, extensive documentation is offered[3], also mentioning Bob's extensibility allowing researchers to use it for their purposes. Finally, the software runs on Linux and MacOS 64-bit operating systems and can be installed with conda or pip. The support for Windows is still planned (Anjos et al., 2012).

Bob (v11.0.0) is used for the experiments declared in this thesis to easily set up a pipeline with the ArcFace-InsightFace model mentioned above for feature extraction. Furthermore, it allows an easy initialization of the described SCface database using a defined protocol and includes tools to efficiently load extracted features using the HDF5 library [4]. In addition, the software offers evaluation metrics helpful for plotting, which allows easy comparison between experiments (e.g., the plots visible in figure 2.2). So, after running the experiments, the recorded evaluation files are handed to Bob to evaluate VP.

[3]https://www.idiap.ch/software/bob/, retrieved July 14, 2022.
[4]https://www.hdfgroup.org/solutions/hdf5/, retrieved July 14, 2022.

3.4 Software SciPy

The thesis at hand, aims to efficiently compare the performance of different similarity functions. There are numerous functions to calculate correlations or distances between lists. Rather than implementing well-known functions from scratch, software providing them is used. SciPy is the perfect candidate for this task for the following reasons:

- SciPy's functionalities are publicly available since the software is open-source.
- It is researcher-friendly and fits the software Bob as it also provides a Python environment.
- NumPy, a Python library supporting rapid vector computations and used for working with extracted features, is extended by SciPy.
- The software connects fast C++ implementations with Python's flexibility, similar to Bob.
- There is a well-maintained and extensive online documentation to help with its functionalities.[5]

The functions important for this thesis are found in the packages `scipy.stats` and `scipy.spatial.distance`. They are listed below, together with their notation used for reference.

1. Correlation functions from `scipy.stats`

• The Spearman correlation:	$S_{spearman}$ (2.11)
• Kendall's tau:	$S_{kendall}$ (2.12)
• The weighted version of Kendall's tau:	$S_{w.kendall}$ (2.14)

2. Distance computations from `scipy.spatial.distance`, where u and v are one-dimensional vectors of size N and $i \in \{1, 2, \ldots, N\}$. Note that the distance measures are negated such that higher scores stand for higher similarity.

   - The cosine similarity:

     $S_{cos}(u, v)$ as defined in formula 2.3

   - The Bray-Curtis similarity:

$$S_{braycurtis}(u,v) = -\frac{\sum |u_i - v_i|}{\sum |u_i + v_i|} \tag{3.1}$$

   - The Canberra similarity:

$$S_{canberra}(u,v) = -\sum \frac{|u_i - v_i|}{|u_i| + |v_i|} \tag{3.2}$$

   - The City Block (Manhattan) similarity:

$$S_{cityblock}(u,v) = -\sum |u_i - v_i| \tag{3.3}$$

   - The squared Euclidean similarity:

$$S_{sq.euclidean}(u,v) = -\|u - v\|_2^2 \tag{3.4}$$

[5] `https://docs.scipy.org/doc/scipy/reference/index.html`, retrieved July 14, 2022.

- The Minkowski similarity, where p defines the order of the norm:

$$S_{minkowski}(u,v) = -\|u-v\|_p = \left(\sum |u_i - v_i|^p\right)^{\frac{1}{p}} \tag{3.5}$$

Note that the squared Euclidean similarity $S_{sq.euclidean}$ (3.4) is monotonous to the Euclidean similarity. However, only the first is considered since the latter computes the square root; therefore, it is slower in execution.

The similarity function $S_{minkowski}$ (3.5) includes the parameter p, which defines the norm of a vector, and is therefore known as the p-norm. $p = 1$ results in the Manhattan norm and equals the City Block distance part of formula (3.3). Similarly, $p = 2$ results in the Euclidean norm and therefore equals the Euclidean distance. As p approaches ∞ the p-norm approaches the maximum norm (3.6). Note that the maximum norm equals the Chebyshev distance. This effect can be further observed in figure 3.3. The most interesting values are given for $0 < p \leq 3$, as the variance is highest.

$$\|u\|_\infty = \max\left(|u_1|, \ldots, |u_N|\right) \tag{3.6}$$

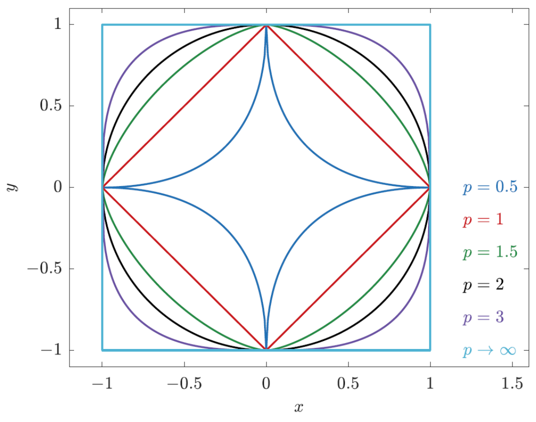


Figure 3.3: UNIT CIRCLES. This figure taken from Mohamad Mezher et al. (2019) illustrates unit circles $(x^p + y^p) = 1$ for various p-values. Unit circles consist of all points, one unit from the circle's center. The figure shows that for instance, the unit circle for $p = 1$ results in a tilted square, which is the unit circle of the Manhattan norm, and for $p = 2$ it resembles a perfect circle, which is the unit circle of the Euclidean norm.

Chapter 4

Approach

This chapter provides an overview of the various methods used for the experiments. First, the term *baseline* is introduced. The first method describes rank list generation to evaluate performance, which is opposed to the direct comparison method. A second method is represented by computing the standard score of the lists l_p and l_g. The implementation details of each section are listed in the appendix A.1.

4.1 Baseline

The baseline describes the direct comparison method on the SCface database. Each probe sample is compared to all gallery samples using the similarity function $S_{cos}(u,v)$ (2.3), where u is defined as the probe's features and v as those of the respective gallery sample. For VP, the resulting scores are evaluated by Bob. The maximum of all scores reveals the gallery image, which is the most similar to the probe image. For IP, their subject IDs are compared to check whether the match is genuine.

4.2 Rank List Method

The baseline compares a low-resolution probe image with a high-resolution gallery image directly. However, as mentioned by Schmidli (2021), many state-of-the-art face recognition algorithms fail this task leaving room for improvement. In light of the cited publications in section 2.4 resulting in more accurate scores, the idea of rank lists is applied as a method in this thesis. The goal is to evaluate whether this method can achieve better low-resolution face recognition the same way it did for the recognition task with different poses.

For the generation of rank lists, a cohort is necessary, which for the SCface database consists of both high **and** low-resolution images covering **all** protocols. Therefore, the cohort is split into samples serving as a reference database for all probe samples and such serving as a reference database for all gallery samples, denoted as probe cohort and gallery cohort, respectively. Since the protocols defined by the database indicate which samples should be compared for a given distance, as mentioned in section 3.1, the probe cohort is filtered using the specified protocol to include only the samples relevant for the current recognition task. The database consists of multiple images per subject (i.e., five different cameras are used). Computing the average of all the subject's features yields the best results (de Freitas Pereira et al., 2022). With this method, only one sample per subject in the cohort serves as a reference, which is an advantage when generating

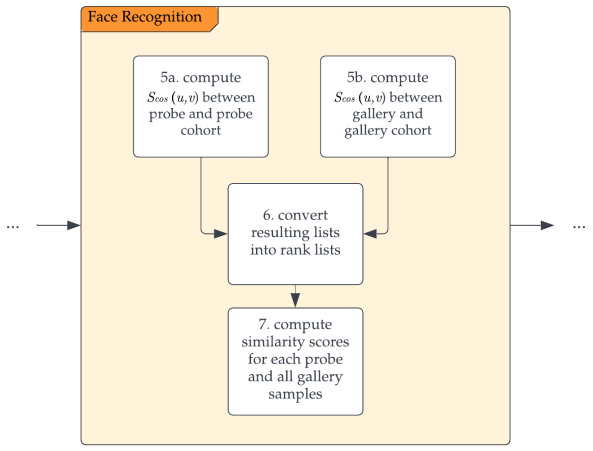

Face Recognition
5a. compute $S_{cos}(u,v)$ between probe and probe cohort
5b. compute $S_{cos}(u,v)$ between gallery and gallery cohort
6. convert resulting lists into rank lists
7. compute similarity scores for each probe and all gallery samples
...
...

Figure 4.1: RANK LIST STAGES FOR SCFACE. This figure shows the order of execution when running the rank list method on the SCface database extending figures 2.1 and 2.4.

rank lists for the probe and gallery samples. Consequently, the resulting rank lists are of equal length, guaranteeing a successful comparison.

Cosine distances are computed between each probe sample and the probe cohort using the distance measure $d_{cos}(u, v)$ as defined in formula (2.4), where u is defined as the probe's features and v as those of the respective cohort sample. This is done after preparing the cohort but before applying the comparison methods and is repeated for each gallery sample and the gallery cohort. Once all distances are calculated, they are saved in a list, which is then turned into the sample's rank list. Rank 0 then indicates the smallest distance (which stands for the highest similarity), rank 1 the second smallest, and so on. Figure 4.1 shows how the process stated in figure 2.4 changes when applied to the SCface database.

Finally, the different comparison methods can be applied, which use the generated rank lists when comparing each probe sample with all gallery samples to compute the similarity scores.

4.3 Standardization Method

The rank list method is useful because the resulting cosine distances significantly differ between the probe and the gallery samples due to the resolution of the images. The similarity between the probe samples and probe cohort is affected more since the images have a low resolution. Therefore, the cosine distances computed between the gallery samples and gallery cohort (which is of high resolution) are more accurate. This issue prohibits an efficient usage of distance computations between the calculated lists of cosine distances l_p for probe and l_g for gallery samples. However, what if this difference in distribution can somehow be neutralized? Then distance computations applied directly on l_p and l_g yield accurate results.

Furthermore, there is no need for rank lists, which might pose an advantage. When generating rank lists, the distribution of the entries is neglected, as the values are assigned an integer rank. Distance computations, on the other hand, consider the gaps between the entries and, therefore, include this distribution. So in summary, the cohort is used, however, without rank lists. To adjust the mentioned difference in distribution between the cosine distances, the lists l_p and l_g are standardized (i.e., the corresponding mean μ is subtracted from each list, and the result is divided by the list's standard deviation σ) as follows, where x stands for l_p and y for l_g.

$$x_z = \frac{x - \mu_x}{\sigma_x} \qquad y_z = \frac{y - \mu_y}{\sigma_y} \tag{4.1}$$

For comparison, only the dividend is applied to l_p and l_g to find out whether dividing by the standard deviation affects the distance computations.

$$x_d = x - \mu_x \qquad y_d = y - \mu_y \tag{4.2}$$

Note that when l_p and l_g are *standardized* (4.1), the functions S_{cos} (2.3) and $S_{sq.euclidean}$ (3.4) become monotonous yielding the same performance results.

Chapter 5

Results

This chapter provides the results of all experiments performed using the rank list and the standardization method on the SCface database. Their performance is compared to that of the baseline using tables for RRs and graphs for the ROC curves. Curves that stick to the upper-left corner stand for accurate performance, (Metz, 1978) and vice versa; the more the curves flatten and move to the lower right, the less precise the corresponding method is. In addition, the CPU runtime is recorded to also compare execution performance.

5.1 Baseline

As mentioned in section 4.1, the baseline describes the direct comparison of probe and gallery images, and its performance serves as an essential reference. As the baseline does not use the cohort, the cohort size does not impact the results. The RR for each protocol and the runtime (measured as mentioned in section A.1.7) is recorded in table 5.1. The ROC curves in figure 5.1 show the baseline's VP. The results show that performance decreases the further away the subjects are recorded. This observation is equally visible for all comparison methods.

	RR (%)		average runtime (*ms*)
close	*medium*	*far*	
100.00	98.18	74.09	382.3718

Table 5.1: BASELINE IP. This table includes the RRs for each protocol given in percentages as well as the average runtime in milliseconds.

5.2 Rank List Method

The rank list method described in section 4.2 uses the cohort. Therefore the results of the small cohort (i.e., consisting of 43 samples) are separated from those of the large cohort (i.e., 86 samples) and presented in the corresponding sections 5.2.2 and 5.2.3. The average time for preprocessing (i.e., generate rank lists) for the small cohort is $pt_s = 503.0539$ milliseconds and $pt_l = 938.4035$ milliseconds for the large cohort.

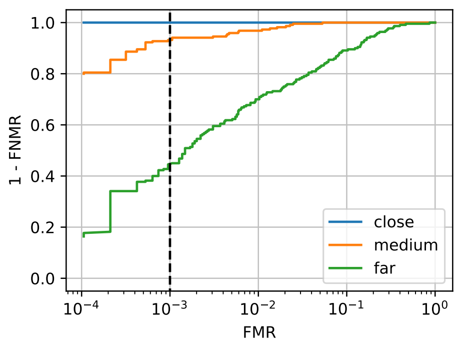


Figure 5.1: BASELINE VP. This figure shows VP of the baseline for every protocol.

The similarity functions $S_{schroff}$ (2.7), $S_{2013mueller}$ (2.8) and $S_{wartmann}$ (2.10) all include parameters, which when changed yield different results. Therefore, the following section provides an overview of the best parameters applied for all consecutive experiments.

5.2.1 Schroff's, Müller's & Wartmann's Parameters

Since the focus is to highlight how different parameters influence the methods' performance, the runtime is not measured for the results presented. Due to the same reasons, only the version using the small cohort is shown. The results for the larger cohort are listed in the appendix A.2.1.

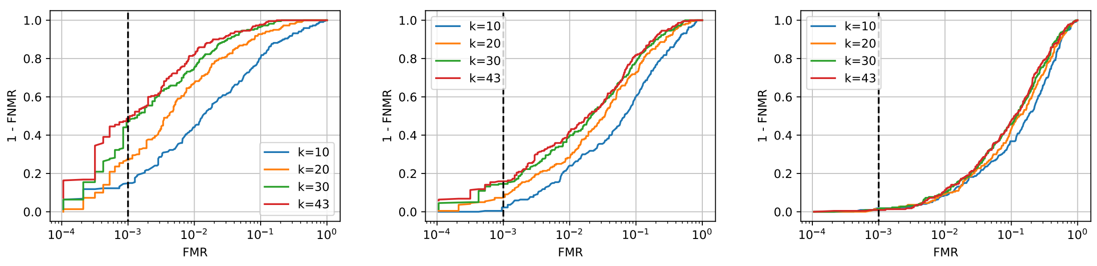
Schroff

The similarity function presented in Schroff et al. (2011) includes the parameter k, which defines up to what point the ranks in the lists are considered (i.e., where similarities are most significant). The small cohort includes 43 samples. Therefore, values $k \in \{10, 20, 30, 43\}$ are presented to show a gradual increase. Table 5.2 summarizes IP. Figure 5.2 shows VP.

The results indicate that (1) a small k is best only for protocol *far* of the small cohort when referring to IP, and (2) otherwise, a large k generates the best IP and VP.

parameter	RR (%)		
	close	*medium*	*far*
$k = 10$	44.55	27.73	**16.36**
$k = 20$	70.00	35.91	**16.36**
$k = 30$	76.36	46.82	13.18
$k = 43$	**83.64**	**50.45**	12.73

Table 5.2: SCHROFF IP SMALL COHORT. This table shows the RRs for each defined k as well as for each protocol given in percentages. The best values for each protocol are marked in **bold**.



(a) *close* (b) *medium* (c) *far*

Figure 5.2: SCHROFF VP SMALL COHORT. This figure including subfigures (a) and (c) shows VP of the similarity function $S_{schroff}$ (2.7) for each protocol using a different parameter k on the small cohort.

Müller 2013

For their similarity function, Müller et al. (2013) make use of a decaying function which includes the parameter $\lambda \in [0.9, 1)$. The values $\lambda \in \{0.90, 0.95, 0.99\}$ are presented in table 5.3 and figure 5.3 to show a gradual increase.

From the results, it becomes clear that (1) a small λ is best only for protocol *far* of the small cohort when referring to IP, and (2) otherwise, a large λ generates the best IP and VP.

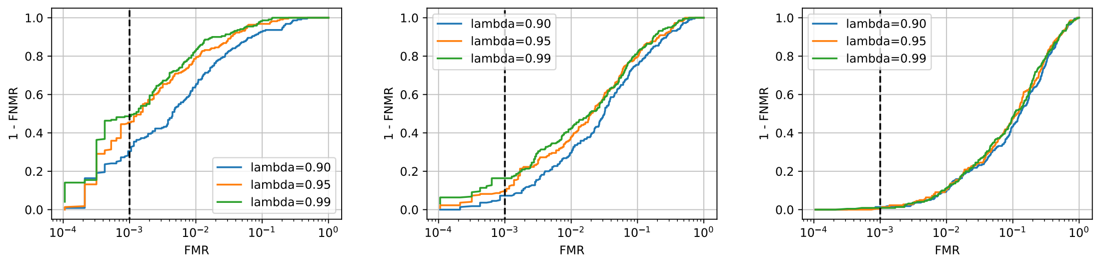
parameter	RR (%)		
	close	*medium*	*far*
$\lambda = 0.90$	70.00	38.64	**17.73**
$\lambda = 0.95$	76.36	43.64	15.91
$\lambda = 0.99$	**81.82**	**50.00**	14.09

Table 5.3: MÜLLER 2013 IP SMALL COHORT. This table shows the RRs for each defined λ as well as for each protocol given in percentages. The best values for each protocol are marked in **bold**.

Wartmann

Wartmann (2021) mentions the values best for the parameters α and β, namely $\alpha = 1.5$ and $\beta = 5$. Applying the similarity function to the low-resolution images of the SCface database, figure 5.4 shows how IP decreases or stagnates for higher values. Table 5.4 lists the most interesting α-β-pairs, and figure 5.5 summarizes VP. Since the parameters are powers of differences in ranks, values slightly larger than one are expected to perform best. The results indicate that

1. α-β-values roughly between one and three result in the most accurate performance,
2. IP is mostly best when both parameters are increased independently;
3. increasing only β has the smallest impact on IP;
4. the parameters overall have very little influence for protocol *far* concerning VP.



(a) *close* (b) *medium* (c) *far*

Figure 5.3: MÜLLER 2013 VP SMALL COHORT. This figure including subfigures (a) - (c) shows VP of the similarity function $S_{2013mueller}$ (2.8) for each protocol using a different parameter λ on the small cohort.

parameter	RR (%) *close*	*medium*	*far*
$\alpha = 1.3, \beta = 1.0$	**86.36**	**49.55**	15.45
$\alpha = 1.4, \beta = 1.0$	85.91	49.09	15.00
$\alpha = 1.5, \beta = 1.0$	**86.36**	49.09	15.45
$\alpha = 1.6, \beta = 1.0$	**86.36**	47.73	16.36
$\alpha = 1.7, \beta = 1.0$	85.91	47.73	16.36
$\alpha = 1.8, \beta = 1.0$	85.00	47.73	16.36
$\alpha = 1.9, \beta = 1.0$	85.00	47.73	**17.27**
$\alpha = 2.0, \beta = 1.0$	85.00	47.27	**17.27**
$\alpha = 2.1, \beta = 1.0$	85.00	45.91	16.82
$\alpha = 2.2, \beta = 1.0$	84.55	45.00	16.82
$\alpha = 2.3, \beta = 1.0$	84.09	44.55	15.91

(a) increasing α

parameter	RR (%) *close*	*medium*	*far*
$\alpha = 1.0, \beta = 2.0$	**86.36**	49.09	15.00
$\alpha = 1.0, \beta = 2.1$	85.91	50.00	15.00
$\alpha = 1.0, \beta = 2.2$	85.45	50.45	15.00
$\alpha = 1.0, \beta = 2.3$	85.45	**51.36**	15.00
$\alpha = 1.0, \beta = 2.4$	85.00	**51.36**	15.00
$\alpha = 1.0, \beta = 2.5$	85.00	50.91	15.00
$\alpha = 1.0, \beta = 2.6$	85.00	50.91	15.00
$\alpha = 1.0, \beta = 2.7$	84.55	50.45	15.45
$\alpha = 1.0, \beta = 2.8$	83.64	50.45	15.45
$\alpha = 1.0, \beta = 2.9$	83.18	50.45	**15.91**
$\alpha = 1.0, \beta = 3.0$	83.18	50.45	15.45

(b) increasing β

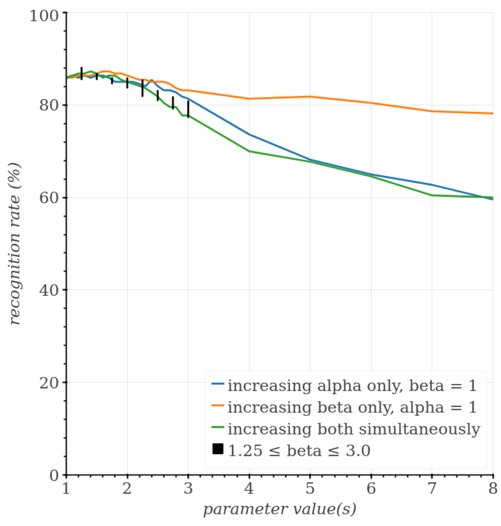
parameter	RR (%) *close*	*medium*	*far*
$\alpha = 1.1, \beta = 1.1$	86.36	**50.91**	15.45
$\alpha = 1.2, \beta = 1.2$	86.82	49.55	15.45
$\alpha = 1.3, \beta = 1.3$	86.82	49.09	15.00
$\alpha = 1.4, \beta = 1.4$	**87.27**	47.73	15.91
$\alpha = 1.5, \beta = 1.5$	86.82	48.64	15.91
$\alpha = 1.6, \beta = 1.6$	85.91	48.18	16.82
$\alpha = 1.7, \beta = 1.7$	86.36	48.64	16.36
$\alpha = 1.8, \beta = 1.8$	86.36	48.18	16.36
$\alpha = 1.9, \beta = 1.9$	85.45	47.73	16.82
$\alpha = 2.0, \beta = 2.0$	85.00	45.45	**17.27**
$\alpha = 2.1, \beta = 2.1$	84.55	45.00	**17.27**

(c) increasing both

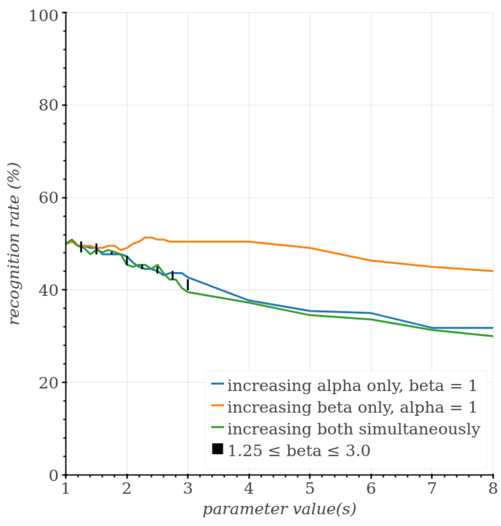
parameter	RR (%) *close*	*medium*	*far*
$\alpha = 1.25, \beta = 1.75$	**88.18**	48.18	15.45
$\alpha = 1.25, \beta = 2.00$	**88.18**	49.09	15.91
$\alpha = 1.25, \beta = 2.25$	87.73	49.55	16.36
$\alpha = 1.25, \beta = 2.50$	87.27	**50.45**	15.91
$\alpha = 1.50, \beta = 1.75$	86.36	47.73	16.82
$\alpha = 1.50, \beta = 2.00$	86.82	49.09	16.36
$\alpha = 1.50, \beta = 2.25$	85.91	48.18	17.27
$\alpha = 1.75, \beta = 2.00$	85.45	47.73	17.27
$\alpha = 1.75, \beta = 2.25$	85.45	47.73	**18.18**
$\alpha = 1.75, \beta = 2.50$	85.45	47.73	**18.18**
$\alpha = 1.75, \beta = 2.75$	85.45	48.18	17.73

(d) various α-β-pairs

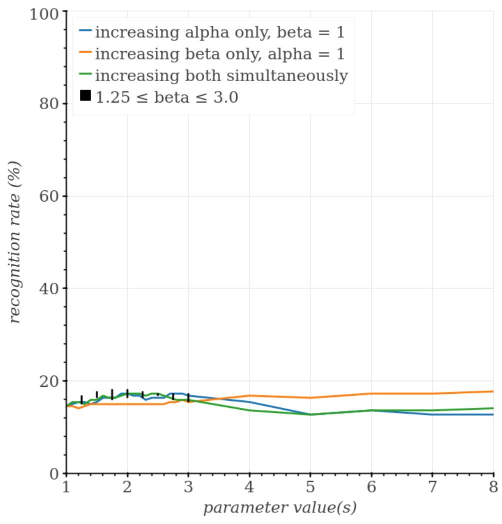
Table 5.4: WARTMANN IP SMALL COHORT. This table including subtables (a) - (d) shows the RRs for each defined α-β-pair as well as for each protocol given in percentages. The best values for each protocol are marked in **bold**.



(a) *close*



(b) *medium*



(c) *far*

Figure 5.4: WARTMANN IP SMALL COHORT. This figure including subfigures (a) - (c) shows IP for various α-β-pairs on the small cohort. The black quads show the range of resulting RRs where α equals the value on the x-axis and $\beta \in [1.25, 3.0]$.

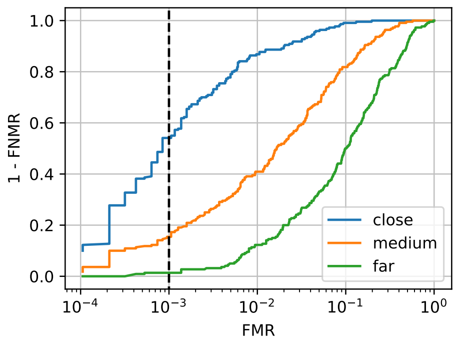


Figure 5.5: WARTMANN VP SMALL COHORT. This figure shows VP of the similarity function $S_{wartmann}$ (2.10). Here $\alpha = 1.0$ and $\beta = 2.8$ for best performance; however, as figure A.3 shows, the values have a small impact.

5.2.2 Small Cohort

This section compares the methods mentioned in section 2.4.2 using the small cohort of 43 samples. For the aforementioned similarity functions, the following parameters are used for all consecutive experiments to maximize performance for protocol *far* (i.e., where the effect of low resolution is the most extreme).

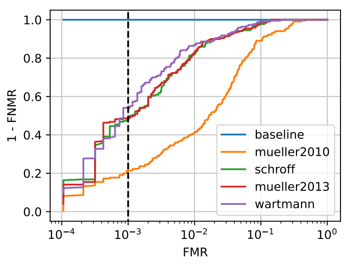
- $S_{schroff}$ (2.7): $k = 20$ for identification and $k = 43$ for verification.
- $S_{2013mueller}$ (2.8): $\lambda = 0.90$ for identification and $\lambda = 0.99$ for verification.
- $S_{wartmann}$ (2.10): $\alpha = 1.75, \beta = 2.25$ for identification and $\alpha = 1.0, \beta = 2.8$ for verification.

Table 5.5 provides all RRs for each protocol and each comparison method. The last column again presents the average runtime for each one. Additionally, for comparison, the identification results of the baseline are given in the last row. Figure 5.6 shows the ROC curves of each method for each protocol. The results make it clear that

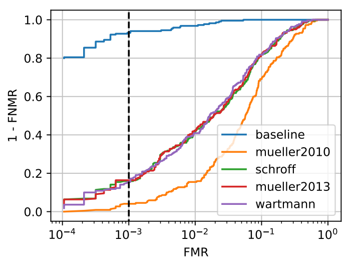
1. by adjusting the parameters, the similarity functions $S_{schroff}$ (2.7), $S_{2013mueller}$ (2.8) and $S_{wartmann}$ (2.10) perform best for protocol *far* regarding IP,
2. for the other protocols, however, weighting and prioritizing ranks is barely superior (i.e., the traditional correlation function $S_{spearman}$ (2.11) performs as one of the best);
3. VP of all similarity functions (except $S_{w.kendall}$ (2.14) and $S_{2010mueller}$ (2.6)) is very similar for all protocols;
4. $S_{wartmann}$ (2.10) slightly performs better with specific parameters, but the difference is minimal;
5. even though $S_{spearman}$ (2.11) is implemented in C++, the function's runtime proves to be the longest;
6. $S_{w.kendall}$ (2.14) does not perform better than the unweighted version.

method	RR (%) *close*	RR (%) *medium*	RR (%) *far*	average runtime (*ms*)
$S_{2010mueller}$ (2.6)	48.64	25.45	15.91	**707.5514**
$S_{schroff}$ (2.7)	70.00	35.91	16.36	838.1356
$S_{2013mueller}$ (2.8)	70.00	38.64	17.73	1302.4865
$S_{wartmann}$ (2.10)	**85.45**	47.73	**18.18**	2143.6471
$S_{spearman}$ (2.11)	83.64	**50.45**	12.73	3426.2422
$S_{kendall}$ (2.12)	81.82	48.64	14.55	3271.7671
$S_{w.kendall}$ (2.14)	68.64	32.73	9.09	1635.2906
baseline	100.00	98.18	74.09	382.3718

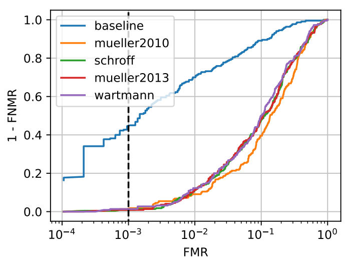
Table 5.5: RANK LIST IP SMALL COHORT. This table includes the RRs of all rank list similarity functions for each protocol given in percentages as well as the average runtime in milliseconds. The best values are marked in **bold**.



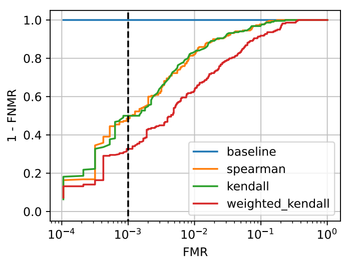
(a) similarity functions, *close*



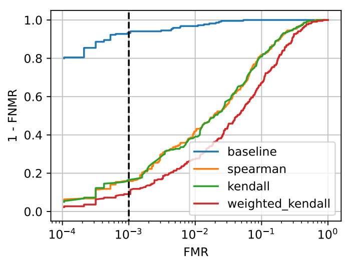
(b) similarity functions, *medium*



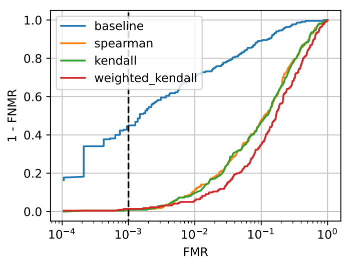
(c) similarity functions, *far*



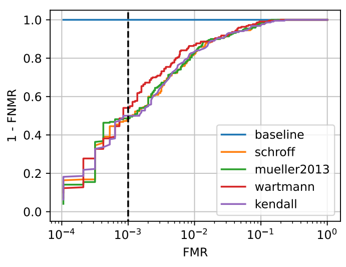
(d) correlation functions, *close*



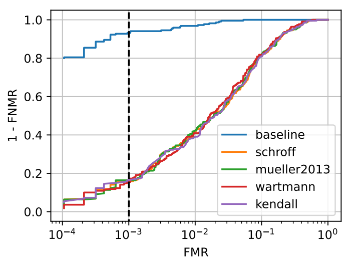
(e) correlation functions, *medium*



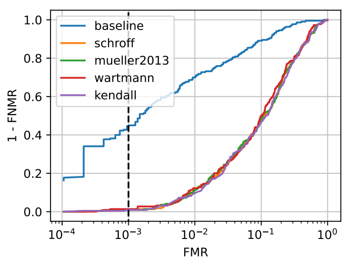
(f) correlation functions, *far*



(g) best combined, *close*



(h) best combined, *medium*



(i) best combined, *far*

Figure 5.6: RANK LIST VP SMALL COHORT. This figure including subfigures (a) - (i) shows VP of all rank list similarity functions for each protocol on the small cohort. Subfigures (a) - (c) show all similarity functions, subfigures (d) - (f) all correlation functions and subfigures (g) - (i) combine the best from both.

5.2.3 Large Cohort

This section presents the results using the large cohort of 86 samples. The values for the parameters as mentioned above in section 5.2.2 change as follows again to maximize performance for protocol *far*:

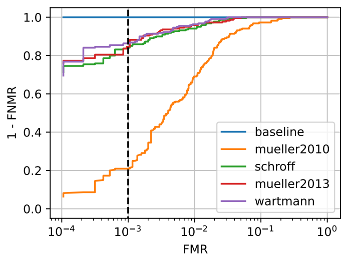
- $S_{schroff}$ (2.7): $k = 86$ for both identification and verification.
- $S_{2013mueller}$ (2.8): $\lambda = 0.99$ for both identification and verification.
- $S_{wartmann}$ (2.10): $\alpha = 1.75, \beta = 2.5$ for identification, $\alpha = 1.0, \beta = 2.8$ for verification remains the same.

Table 5.6 again provides the results of IP and figure 5.7 those of VP. The results indicate that

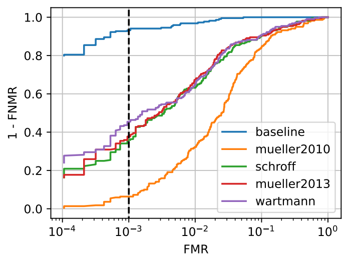
1. in comparison to the small cohort, the similarity functions $S_{schroff}$ (2.7), $S_{2013mueller}$ (2.8), $S_{wartmann}$ (2.10) and $S_{spearman}$ (2.11) perform very similar for all protocols regarding IP (i.e., weighting and prioritizing ranks is barely superior for any protocol);
2. all functions (except $S_{w.kendall}$ (2.14) and $S_{2010mueller}$ (2.6)) again generate very similar results for all protocols regarding VP;
3. $S_{spearman}$ (2.11) again takes a long time to complete;
4. $S_{kendall}$ (2.12) again proves better performance than the weighted version;
5. when compared to the small cohort, the RRs and ROC curves are more accurate overall,
6. whereas the runtime increases for each similarity function.

method	RR (%) *close*	*medium*	*far*	average runtime (*ms*)
$S_{2010mueller}$ (2.6)	72.73	46.36	17.27	**1271.3769**
$S_{schroff}$ (2.7)	**97.27**	71.36	27.27	1729.9866
$S_{2013mueller}$ (2.8)	**97.27**	**73.64**	27.27	2694.7465
$S_{wartmann}$ (2.10)	96.82	72.27	**28.64**	4309.2037
$S_{spearman}$ (2.11)	**97.27**	71.36	27.27	3919.1914
$S_{kendall}$ (2.12)	96.82	72.27	26.82	3740.4091
$S_{w.kendall}$ (2.14)	84.55	48.18	18.18	2953.2181
baseline	100.00	98.18	74.09	382.3718

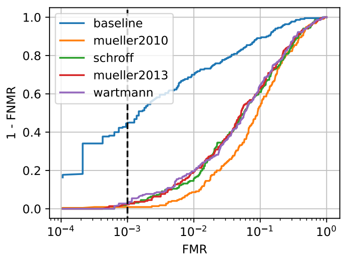
Table 5.6: RANK LIST IP LARGE COHORT. This table includes the RRs of all rank list similarity functions for each protocol given in percentages as well as the average runtime in milliseconds. The best values are marked in **bold**.



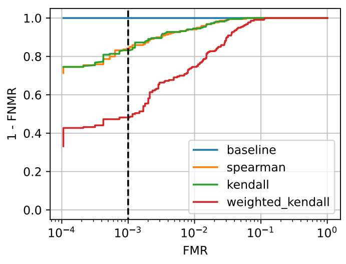
(a) similarity functions, *close*



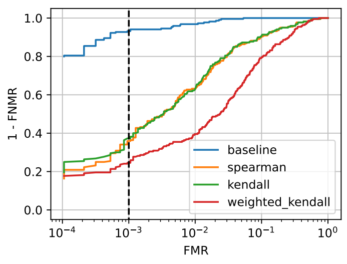
(b) similarity functions, *medium*



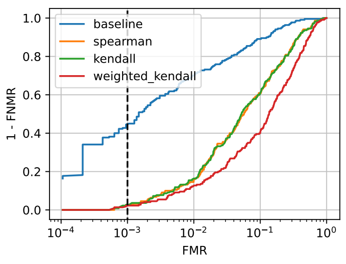
(c) similarity functions, *far*



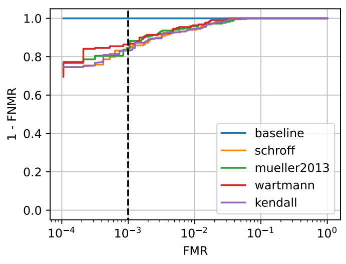
(d) correlation functions, *close*



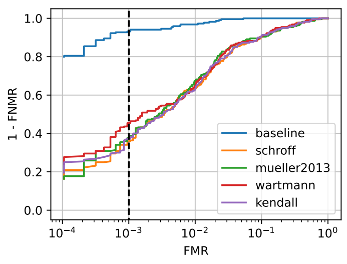
(e) correlation functions, *medium*



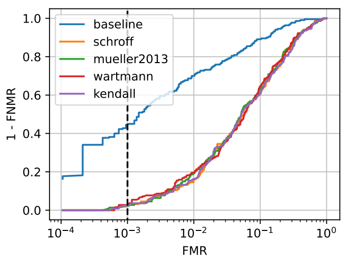
(f) correlation functions, *far*



(g) best combined, *close*



(h) best combined, *medium*



(i) best combined, *far*

Figure 5.7: RANK LIST VP LARGE COHORT. This figure including subfigures (a) - (i) shows VP of all rank list similarity functions for each protocol on the large cohort. Subfigures (a) - (c) show all similarity functions, subfigures (d) - (f) all correlation functions and subfigures (g) - (i) combine the best from both.

5.3 Standardization Method

Similarly to section 5.2, the results of the small cohort are separated from those of the large cohort for the standardization method. The average preprocessing time pt (i.e. for the methods *standardize* (4.1), *subtract_mean* (4.2) and *omitted*) is listed below:

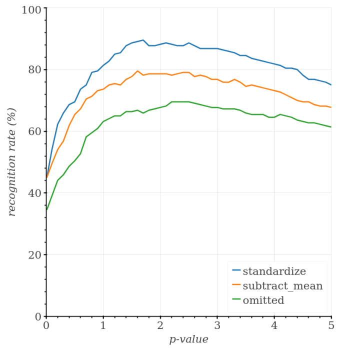
preprocessing method	small cohort s (ms)	large cohort l (ms)
standardize (4.1): std	$pt_{sstd} = 504.9220$	$pt_{lstd} = 977.8654$
subtract_mean (4.2): sm	$pt_{ssm} = 512.3031$	$pt_{lsm} = 984.9502$
omitted: om	$pt_{som} = 525.6306$	$pt_{lom} = 995.9936$

Surprisingly, the measurements for no preprocessing are the longest. There is no obvious explanation why omitting preprocessing, as described in section A.1.5, does not result in the shortest time, even though fewer computations are executed.

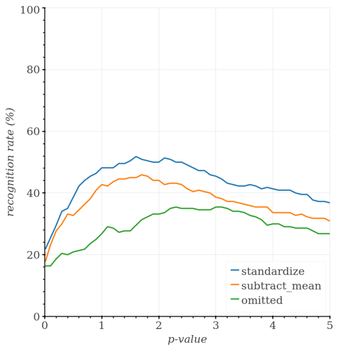
5.3.1 Minkowski's Parameter

Figure 5.8 shows how IP decreases or stagnates for higher values. Table 5.7 lists the most interesting p-values, and figure 5.9 depicts the ROC curves. The appendix A.2.2 includes the results for the large cohort, since here only those of the small cohort are presented for equal reasons as described in section 5.2.1. The results show that

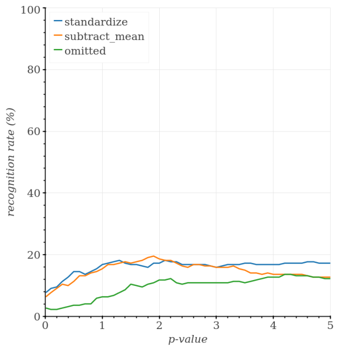
1. $p = 2.0$ yields the best VP (no matter the preprocessing method);
2. for protocol *far* the *subtract_mean* method (4.2) with minimal difference produces the best IP,
3. otherwise, the *standardize* method (4.1) is more accurate;
4. values around $p = 1.7$ show the most precise performance regarding IP.



(a) *close*

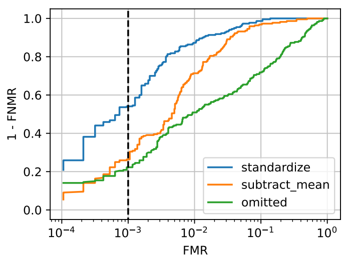


(b) *medium*

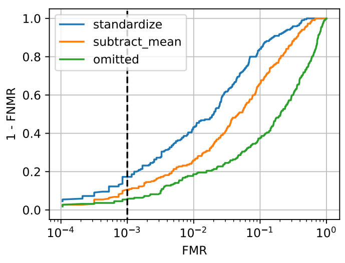


(c) *far*

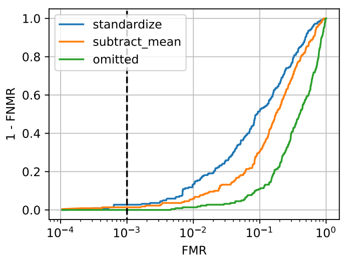
Figure 5.8: MINKOWSKI IP SMALL COHORT. This figure including subfigures (a) - (c) shows IP for various p-values on the small cohort.



(a) *close*



(b) *medium*



(c) *far*

Figure 5.9: MINKOWSKI VP SMALL COHORT. This figure including subfigures (a) - (c) shows VP of the similarity function $S_{minkowski}$ (3.5). Here $p = 2.0$ for best performance (i.e., same VP as $S_{sq.euclidean}$ (3.4)), however as figure A.7 shows, the value has a small impact.

parameter	RR (%)		
	close	*medium*	*far*
$p = 1.3$	85.45	49.55	**18.18**
$p = 1.4$	87.73	49.55	17.27
$p = 1.5$	88.64	50.45	16.82
$p = 1.6$	89.09	**51.82**	16.82
$p = 1.7$	**89.55**	50.91	16.36
$p = 1.8$	87.73	50.45	15.91
$p = 1.9$	87.73	50.00	17.27
$p = 2.0$	88.18	50.00	17.27
$p = 2.1$	88.64	51.36	**18.18**
$p = 2.2$	88.18	50.91	17.73
$p = 2.3$	87.73	50.00	17.73

(a) *standardize*

parameter	RR (%)		
	close	*medium*	*far*
$p = 1.3$	75.00	44.55	17.27
$p = 1.4$	76.82	44.55	17.73
$p = 1.5$	77.73	45.00	17.27
$p = 1.6$	**79.55**	45.00	17.73
$p = 1.7$	78.18	**45.91**	18.18
$p = 1.8$	78.64	45.45	19.09
$p = 1.9$	78.64	44.09	**19.55**
$p = 2.0$	78.64	44.09	18.64
$p = 2.1$	78.64	42.73	18.18
$p = 2.2$	78.18	43.18	18.18
$p = 2.3$	78.64	43.18	17.27

(b) *subtract_mean*

parameter	RR (%)		
	close	*medium*	*far*
$p = 2.0$	67.73	33.18	11.82
$p = 2.1$	68.18	33.64	11.82
$p = 2.2$	**69.55**	35.00	12.27
$p = 2.3$	**69.55**	**35.45**	10.91
$p = 2.4$	**69.55**	35.00	10.45
$p = 2.5$	**69.55**	35.00	10.91
$p = 2.6$	69.09	35.00	10.91
$p = 2.7$	68.64	34.55	10.91
$p = 2.8$	68.18	34.55	10.91
$p = 2.9$	67.73	34.55	10.91
$p = 3.0$	67.73	35.45	10.91
$p = 4.2$	65.00	29.09	**13.64**

(c) *omitted*

Table 5.7: MINKOWSKI IP SMALL COHORT. This table including subtables (a) - (c) shows the RRs for each defined p and preprocessing method as well as for each protocol given in percentages. The best values for each protocol are marked in **bold**.

5.3.2 Small Cohort

This section compares the methods mentioned in section 3.4 using the small cohort of 43 samples. Table 5.8 provides all RRs for each protocol and each comparison method with each preprocessing method. To maximize the IP of protocol *far* for $S_{minkowski}$ (3.5),

- $p = 2.1$ for the *standardize* preprocessing method (4.1),
- $p = 1.9$ for the *subtract_mean* method (4.2),
- and $p = 4.2$ for no preprocessing.

Figure 5.10 shows the ROC curves of each method for each protocol, including the one from the baseline. From the results, it is clear that

1. for the *omitted* preprocessing method, both IP and VP are worse for every distance computation (except S_{cos} (2.3)) when compared to the *standardize* (4.1) and *subtract_mean* (4.2) methods,
2. this is also true for $S_{canberra}$ (3.2);
3. even though S_{cos} (2.3) takes the longest to execute,
4. it shows very solid performance overall, yielding the same results for both the *standardize* (4.1) and *subtract_mean* (4.2) methods;
5. for those preprocessing methods $S_{braycurtis}$ (3.1) shows equal VP,
6. whereas all the other distance computations generate different ROC curves;
7. $S_{cityblock}$ (3.3) has the shortest runtime for all preprocessing methods, closely followed by $S_{sq.euclidean}$ (3.4) and $S_{minkowski}$ (3.5);
8. the *subtract_mean* preprocessing method (4.2) is only best for the protocol *far* regarding $S_{sq.euclidean}$ (3.4) and $S_{minkowski}$ (3.5),
9. otherwise the *standardize* method (4.1) proves better results.

method	RR (%) *close*	*medium*	*far*	average runtime (*ms*)
S_{cos} (2.3)	88.18	50.00	17.27	768.0994
$S_{braycurtis}$ (3.1)	83.64	48.18	15.91	610.4427
$S_{canberra}$ (3.2)	63.18	33.64	12.73	760.6298
$S_{cityblock}$ (3.3)	81.36	48.18	16.82	**574.8153**
$S_{sq.euclidean}$ (3.4)	88.18	50.00	17.27	590.9191
$S_{minkowski}$ (3.5)	**88.64**	**51.36**	**18.18**	596.6574
baseline	100.00	98.18	74.09	382.6218

(a) *standardize*

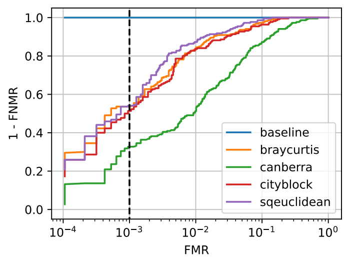
method	RR (%) *close*	*medium*	*far*	average runtime (*ms*)
S_{cos} (2.3)	**88.18**	**50.00**	17.27	817.9292
$S_{braycurtis}$ (3.1)	84.09	48.64	15.00	602.8524
$S_{canberra}$ (3.2)	63.64	34.09	13.18	771.1784
$S_{cityblock}$ (3.3)	73.64	42.73	15.45	**580.9109**
$S_{sq.euclidean}$ (3.4)	78.64	44.09	18.64	596.1222
$S_{minkowski}$ (3.5)	78.64	44.09	**19.55**	591.9678
baseline	100.00	98.18	74.09	382.6218

(b) *subtract_mean*

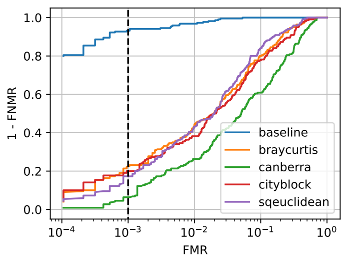
method	RR (%) *close*	*medium*	*far*	average runtime (*ms*)
S_{cos} (2.3)	**78.64**	**43.64**	**18.64**	844.0956
$S_{braycurtis}$ (3.1)	62.73	26.36	6.82	626.2914
$S_{canberra}$ (3.2)	64.55	28.18	6.36	765.5324
$S_{cityblock}$ (3.3)	63.18	26.82	6.36	**576.3784**
$S_{sq.euclidean}$ (3.4)	67.73	33.18	11.82	677.6718
$S_{minkowski}$ (3.5)	65.00	29.09	13.64	594.4734
baseline	100.00	98.18	74.09	382.6218

(c) *omitted*

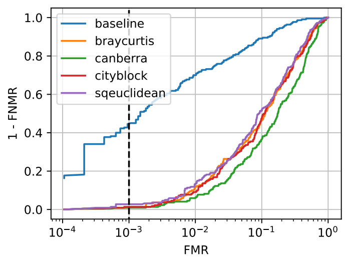
Table 5.8: STANDARDIZATION IP SMALL COHORT. This table including subtables (a) - (c) lists the RRs of all standardization similarity functions for each protocol given in percentages as well as the average runtime in milliseconds for each preprocessing method. The best values are marked in **bold**.



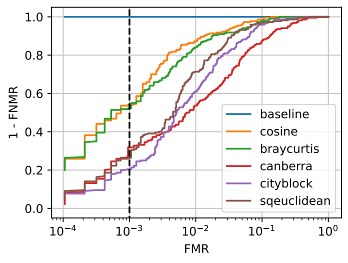
(a) *standardize, close*



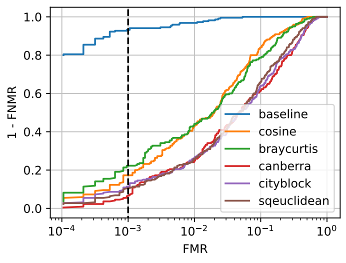
(b) *standardize, medium*



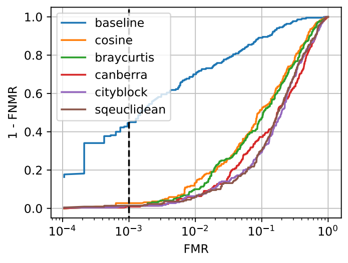
(c) *standardize, far*



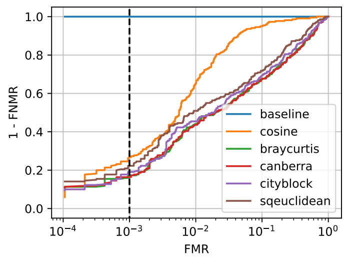
(d) *subtract_mean, close*



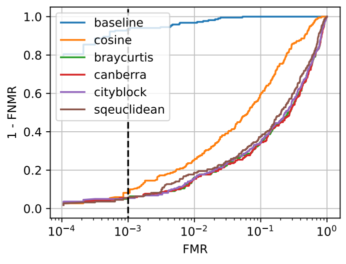
(e) *subtract_mean, medium*



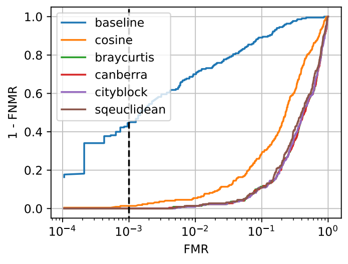
(f) *subtract_mean, far*



(g) *omitted, close*



(h) *omitted, medium*



(i) *omitted, far*

Figure 5.10: STANDARDIZATION VP SMALL COHORT. This figure including subfigures (a) - (i) shows VP of all standardization similarity functions for each protocol on the small cohort for each preprocessing method. Subfigures (a) - (c) show the *standardize* (4.1), subfigures (d) - (f) the *subtract_mean* (4.2) and subfigures (g) - (i) the *omitted* processing method. Note that $S_{minkowski}$ (3.5) is equal to $S_{sq.euclidean}$ (3.4) and, therefore, left away.

5.3.3 Large Cohort

This section presents the results using the large cohort of 86 samples. Again, to maximize the IP of protocol *far* for $S_{minkowski}$ (3.5),

- $p = 2.6$ for the *standardize* preprocessing method (4.1),
- $p = 1.4$ for the *subtract_mean* method (4.2),
- and $p = 4.2$ for no preprocessing.

Table 5.9 again provides the results of IP and figure 5.11 those of VP. The results indicate that

1. in comparison to the small cohort, for the *omitted* preprocessing method, both IP and VP are now worse for every distance computation (also for S_{cos} (2.3)) when compared to the *standardize* (4.1) and *subtract_mean* (4.2) methods,
2. again this is also true for $S_{canberra}$ (3.2);
3. S_{cos} (2.3) again takes the longest,
4. but shows very solid performance;
5. the *standardize* (4.1) method proves the best performance results;
6. $S_{cityblock}$ (3.3) is only fastest for the *standardize* (4.1) method,
7. otherwise $S_{minkowski}$ (3.5) proves a very short execution time,
8. directly followed by $S_{sq.euclidean}$ (3.4);
9. when compared to the small cohort, the RRs and ROC curves are more accurate overall,
10. whereas the runtime increases for each similarity function.

method	RR (%)			average runtime (*ms*)
	close	*medium*	*far*	
S_{cos} (2.3)	**97.73**	**74.55**	27.73	1199.9063
$S_{braycurtis}$ (3.1)	97.27	71.82	26.36	1087.3100
$S_{canberra}$ (3.2)	86.36	56.36	16.82	1238.3367
$S_{cityblock}$ (3.3)	97.27	68.64	25.45	**1070.965**
$S_{sq.euclidean}$ (3.4)	**97.73**	**74.55**	27.73	1076.1242
$S_{minkowski}$ (3.5)	95.91	72.73	**29.55**	1042.3849
baseline	100.00	98.18	74.09	382.6218

(a) *standardize*

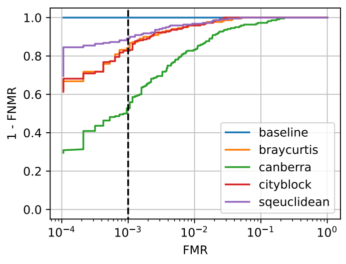
method	RR (%)			average runtime (*ms*)
	close	*medium*	*far*	
S_{cos} (2.3)	**97.73**	**74.55**	27.73	1299.6512
$S_{braycurtis}$ (3.1)	97.27	71.82	25.45	1064.4419
$S_{canberra}$ (3.2)	85.91	57.73	17.27	1237.2778
$S_{cityblock}$ (3.3)	93.18	67.27	27.27	1078.6695
$S_{sq.euclidean}$ (3.4)	94.55	71.82	26.36	1079.5351
$S_{minkowski}$ (3.5)	95.00	70.00	**29.09**	**1040.7184**
baseline	100.00	98.18	74.09	382.6218

(b) *subtract_mean*

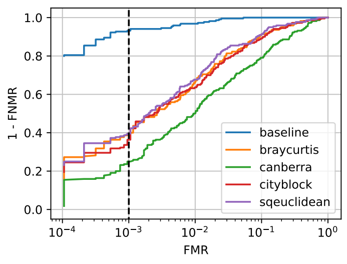
method	RR (%)			average runtime (*ms*)
	close	*medium*	*far*	
S_{cos} (2.3)	**94.55**	**70.91**	**26.36**	1329.5586
$S_{braycurtis}$ (3.1)	79.55	40.91	6.36	1075.9546
$S_{canberra}$ (3.2)	79.55	41.52	6.82	1222.0306
$S_{cityblock}$ (3.3)	78.18	40.00	6.36	1082.0953
$S_{sq.euclidean}$ (3.4)	83.18	46.36	13.64	1081.1378
$S_{minkowski}$ (3.5)	81.36	43.64	16.36	**1071.235**
baseline	100.00	98.18	74.09	382.6218

(c) *omitted*

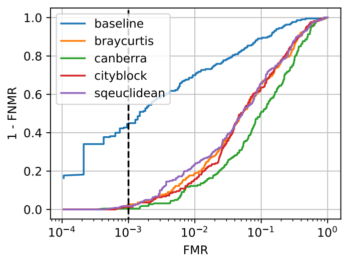
Table 5.9: STANDARDIZATION IP LARGE COHORT. This table including subtables (a) - (c) lists the RRs of all standardization similarity functions for each protocol given in percentages as well as the average runtime in milliseconds for each preprocessing method. The best values are marked in **bold**.



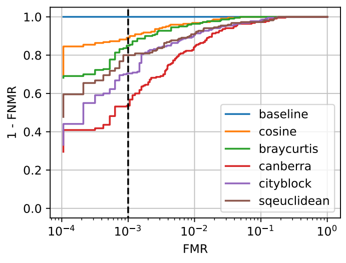
(a) *standardize, close*



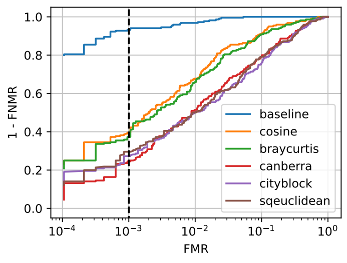
(b) *standardize, medium*



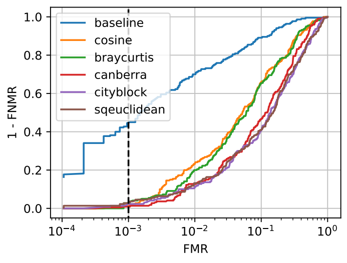
(c) *standardize, far*



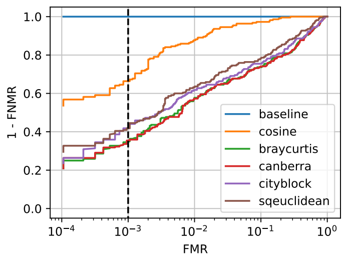
(d) *subtract_mean, close*



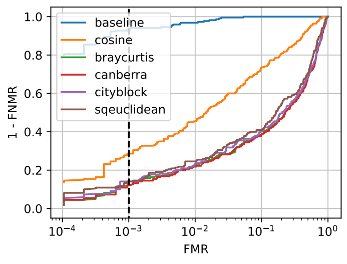
(e) *subtract_mean, medium*



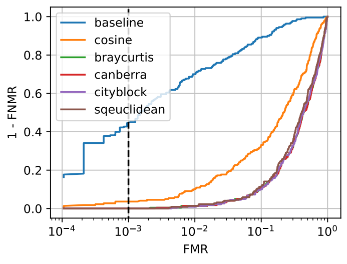
(f) *subtract_mean, far*



(g) *omitted, close*



(h) *omitted, medium*



(i) *omitted, far*

Figure 5.11: Standardization VP Large Cohort. This figure including subfigures (a) - (i) shows VP of all standardization similarity functions for each protocol on the large cohort for each preprocessing method. Subfigures (a) - (c) show the *standardize* (4.1), subfigures (d) - (f) the *subtract_mean* (4.2) and subfigures (g) - (i) the *omitted* processing method. Note that $S_{minkowski}$ (3.5) is equal to $S_{sq.euclidean}$ (3.4) and, therefore, left away.

Chapter 6

Discussion

First, considering $S_{schroff}$ (2.7) intuitively, one might think that considering all ranks (i.e., choosing the largest k) should always yield the best results since more data is considered. As mentioned in section 5.2.1, this is true for the large cohort but not entirely for the small cohort. This behavior might follow due to the small number of images that, in addition, are of the lowest resolution (i.e., protocol *far*). Therefore, a big part of the features might rather introduce noise than lead to more accurate performance. In this case, it is beneficial only to consider the most significant ranks and ignore the rest (i.e., choosing a small k).

$S_{2013mueller}$ (2.8) shows similar behavior. Higher values for λ weight low ranks less than the lowest λ-value of 0.90. As mentioned in section 5.2.1, once the cohort is big enough, the highest λ shows the best performance. But when dealing with a limited cohort size of 43 samples, the lowest λ-value is best. This behavior is consistent with the one of $S_{schroff}$ (2.7). For protocol *far*, the most significant ranks are weighted maximally to exclude higher ranks, which might disrupt the performance as much as possible.

It follows that weighting ranks like in $S_{2013mueller}$ (2.8) or only considering the most significant ones like in $S_{schroff}$ (2.7) is beneficial when dealing with smaller cohort sizes (of very low resolution) since the parameters can be adjusted. By only considering the most significant ranks, their performance becomes more accurate than similarity functions which consider all ranks, therefore, also including noise (e.g., $S_{spearman}$ (2.11)). On the other hand, with a large cohort, the most significant ranks consist of more than just a few, which should all be considered. Hence, the weight should be reduced and k increased. This is coherent with the results.

In contrast, the weighted version of Kendall's tau $S_{w.kendall}$ (2.14) never outperforms $S_{kendall}$ (2.12). However, this is not necessarily the normal case, as various weighting functions can be used in $S_{w.kendall}$ (2.14). The weighting function described in section 2.4.2 does weight low ranks more than higher ones, however similarly to $S_{2010mueller}$ (2.6) it is not decaying fast enough to show an advantage over the unweighted version. Applying a weighting function similar to the one used in $S_{2013mueller}$ (2.8) might yield better performance.

As mentioned in section 5.2.2, $S_{spearman}$ (2.11) takes a long time to execute compared to the other functions. Due to its functionality of calculating **all** differences between the entries of two lists, this is not surprising. Here, $S_{schroff}$ (2.7) shows an advantage, as it only includes simple computations and uses an adjustable parameter. The function stays monotonous to $S_{spearman}$ (2.11) but can guarantee a faster execution, which is coherent with the results.

As mentioned in section 4.3, there is a difference in distribution for the lists l_p and l_g due to image resolution. Therefore, directly applying distance computations without neutralizing this difference results in inaccurate performance. The results confirm this assumption since *omitting* preprocessing shows the worst performance. S_{cos} (2.3) alone, as listed in sections 5.3.2 and 5.3.3, is the similarity function not significantly affected by the preprocessing method. Its stability is given by how the similarity is computed. Since it is represented using the dot product of two vectors and their magnitude, the result is normalized and, therefore, not heavily affected by the difference in preprocessing. On the other hand, the results also indicate that S_{cos} (2.3) is the slowest in execution. Due to simplicity, the cosine distance provided by SciPy is used; however, the computation could be manually simplified by first normalizing both vectors (i.e., dividing them through their corresponding magnitude) and only then calculating the dot product. This change in execution order guarantees a shorter runtime.

The VP of $S_{braycurtis}$ (3.1) remains similar for both preprocessing methods due the same reasons as for S_{cos} (2.3). For each preprocessing method, the divisor remains a normalization factor, therefore, not affecting the ROC curves.

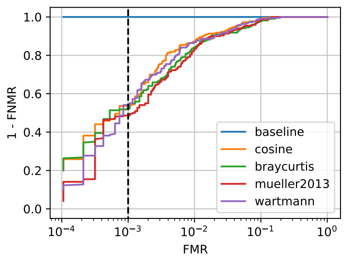
$S_{canberra}$ (3.2) shows very interesting results. The similarity function weights low values the most. When applying the *standardize* (4.1) and *subtract_mean* (4.2) preprocessing methods, the function contrarily weights intermediate values the most since those are now closer to zero. Intuitively, the results should be worse than when *omitting* preprocessing; however, the experiments show otherwise. There is no obvious explanation for this behavior, and it might indicate that weighting intermediate values the most yields better performance in general. Further research is necessary to be certain.

Since the difference between the *standardize* (4.1) and the *subtract_mean* (4.2) preprocessing method only includes a divisor, with which the results are normalized, the performance is expected to not differ to a large extent. The results are consistent with this assumption; however, the first method shows slightly better performance overall.

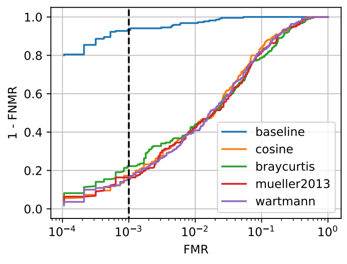
When comparing rank list comparison methods to those of standardization, the best results for IP and VP are mostly very similar for all protocols and either cohort size. $S_{minkowski}$ (3.5) proves the best results for protocol *far*. It profits from a relatively simple implementation, similar to $S_{sq.euclidean}$ (3.4) and, therefore, is expected to have a similar runtime. In addition, the parameter is adjustable, which enables the performance to be maximized. This is coherent with the results. Furthermore, significant differences can be seen in all the functions' runtimes. Distance computations execute much faster than any of the rank list methods. This difference can be explained due to the implementation language. The distance computations provided by SciPy are implemented in C++, whereas the rank list methods are implemented in Python. C++ is the faster language when it comes to calculations like these. Therefore, it is unclear whether the distance computations are, in fact, faster than the rank list methods. Tables 6.1 and 6.2, as well as figures 6.1 and 6.2 provide a summarized overview of both methods combined for a better comparison.

method	RR (%) *close*	RR (%) *medium*	RR (%) *far*	average runtime (*ms*) *pt* excl.
$S_{wartmann}$ (2.10)	85.45	47.73	18.18	2143.6471
$S_{spearman}$ (2.11)	83.64	**50.45**	12.73	3426.2422
S_{cos} (2.3)	**88.18**	50.00	17.27	768.0994
$S_{sq.euclidean}$ (3.4)	78.64	44.09	18.64	**590.9191**
$S_{minkowski}$ (3.5)	78.64	44.09	**19.55**	596.6574
baseline	100.00	98.18	74.09	382.3718

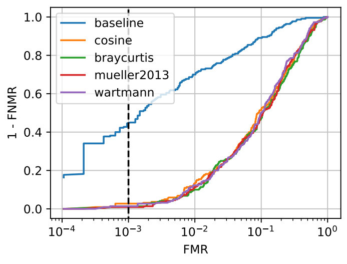
Table 6.1: BEST IP SMALL COHORT. This table includes the RRs for each protocol given in percentages as well as the average runtime in milliseconds. The best values are marked in **bold**. The following adjustments lead to the best IP for protocol *far*: for $S_{wartmann}$ (2.10) $\alpha = 1.75, \beta = 2.25$; $p = 1.9$ for $S_{minkowski}$ (3.5); and for the standardization method the preprocessing *subtract_mean* (4.2) is applied.



(a) best combined, *close*



(b) best combined, *medium*

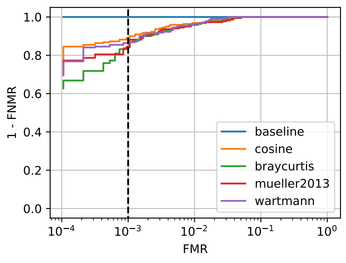


(c) best combined, *far*

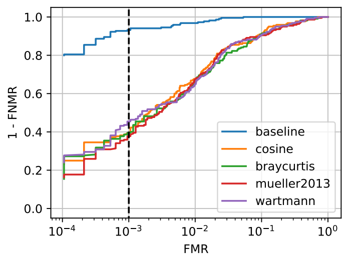
Figure 6.1: BEST VP SMALL COHORT. This figure including subfigures (a) - (c) shows VP for each protocol on the small cohort. The following adjustments lead to the best VP: for $S_{wartmann}$ (2.10) $\alpha = 1.0, \beta = 2.8$; for $S_{2013mueller}$ (2.8) $\lambda = 0.99$; and for the standardization method the preprocessing *standardize* (4.1) is applied.

method	RR (%) *close*	RR (%) *medium*	RR (%) *far*	average runtime (*ms*) *pt* excl.
$S_{wartmann}$ (2.10)	96.82	72.27	28.64	4309.2037
$S_{spearman}$ (2.11)	97.27	71.36	27.27	3919.1914
S_{cos} (2.3)	**97.73**	**74.55**	27.73	1299.6512
$S_{sq.euclidean}$ (3.4)	**97.73**	**74.55**	27.73	1079.5351
$S_{minkowski}$ (3.5)	95.91	72.73	**29.55**	**1040.7184**
baseline	100.00	98.18	74.09	382.3718

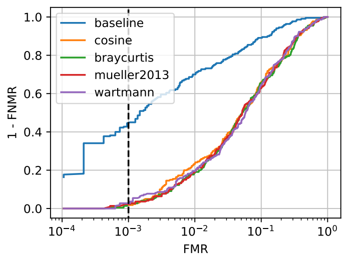
Table 6.2: BEST IP LARGE COHORT. This table includes the RRs for each protocol given in percentages as well as the average runtime in milliseconds. The best values are marked in **bold**. The following adjustments lead to the best IP for protocol *far*: for $S_{wartmann}$ (2.10) $\alpha = 1.75, \beta = 2.5$; $p = 2.6$ for $S_{minkowski}$ (3.5); and for the standardization method the preprocessing *standardize* (4.1) is applied.



(a) best combined, *close*



(b) best combined, *medium*



(c) best combined, *far*

Figure 6.2: BEST VP LARGE COHORT. This figure including subfigures (a) - (c) shows VP for each protocol on the large cohort. The following adjustments lead to the best VP: for $S_{wartmann}$ (2.10) $\alpha = 1.0, \beta = 2.8$; for $S_{2013mueller}$ (2.8) $\lambda = 0.99$; and for the standardization method the preprocessing *standardize* (4.1) is applied.

As mentioned in sections 5.2.3 and 5.3.3, performance and runtime increase for a larger cohort size. Due to the more significant number of samples which need to be considered, the increase in runtime is not surprising. As visible in figure 2.2, the values are expected to follow a standard normal distribution. The theory of the standard normal distribution for high N states that the higher the number of values is, the more accurate the distribution becomes. This is coherent with the results because a larger cohort leads to better performance. However, since it is only an assumption that the values follow a standard normal distribution, the distribution might show a skewness instead. Therefore, more accurate results might be achieved by weighting the values according to their position in the distribution, such that a possible skewness is included.

Even after discussing all the aspects above, the obvious is visible throughout all tables and figures. The baseline comparison, which directly compares probe with gallery samples, still has the best IP and VP for all protocols and either cohort size. In addition, the baseline remains the fastest in execution. This is to be expected as less preprocessing is necessary.

Finally, the results presented are limited by several factors. The first and most straightforward one is the cohort size. As mentioned, the larger the cohort, the more accurate the performance becomes. However, the SCface database tailored to the experiments at hand does not support a cohort more extensive than the 86 samples.

The second factor is presented by the CNN model used for feature extraction. The applied ArcFace-InsightFace-IResNet100 model generally shows outstanding performance, as shown by Deng et al. (2019); however, it might not be the best suitable one for low-resolution images.

Last, the SCface database mainly includes male subjects between the ages of 21 and 23, as mentioned in section 3.1. Due to the small number of female subjects, this might introduce a particular bias in the results as comparing males with females might return a different similarity as a male with male or a female with female subject comparison. The same applies to the age distribution (see figure 3.2) as comparing subjects of older age might return a different similarity as subjects of younger age or a young with old subject comparison.

Chapter 7

Conclusion

The work at hand aims to set side-by-side various face recognition comparison methods by evaluating their performance. For this, the reader is first introduced to face recognition in general. The recognition process is presented by explaining the various stages completed when running the recognition task. The idea of using a reference database (i.e., the cohort) is described. Different rank list comparison methods introduced by previous researchers are listed. Next, chapter 3 provides information about the tools and software used for running the experiments, followed by a description of the term *baseline* and how the rank list and standardization methods are applied with which the results in chapter 5 are generated.

The results show that for both the standardization and the rank list method, performance continuously decreases the further away the subject is recorded (i.e., from protocol *close* to protocol *far*). Furthermore, both methods perform worse using the smaller cohort, clearly highlighting the advantage of a larger cohort. All results distinctly indicate, however, that the usage of a cohort overall still needs further improvement as the baseline performance remains the best for comparing low-resolution probe with high-resolution gallery images.

Future research should focus on several aspects. For one, results might be further improved with an even larger cohort. However, how large the cohort must be to reach the baseline's performance is unknown. Theoretically, the results can be extrapolated to estimate the minimum cohort size after running the experiments with various cohort sizes. Still, the baseline's performance might not even be reached with the largest cohort since the approach of using a reference database might show certain bottlenecks leading to stagnation of performance improvement. Therefore, to be sure, future research should evaluate larger cohort sizes.

Second, to be able to tell whether the standardization method is genuinely faster than the rank list method, both functions must be implemented in the same programming language, preferably C++. Only then can the runtimes be compared efficiently.

Third, the distribution of the computed values should be examined, since it is only assumed, that it represents a standard normal distribution. In doing so, a possible skewness can either be excluded or taken into account properly.

Next, as mentioned, the ArcFace-InsightFace-IResNet100 model might not be best for feature extraction of low-resolution images. Therefore, future research should extract features using various CNN models, which are then used for comparison and might yield better performance. In a further step, the best performance might even be reached by using a different model for feature extraction of the high-resolution images as for features extraction of low-resolution images.

In any case, for meaningful results, future research should focus on generating larger datasets of low-resolution images. The databases should be made publicly available and should not lack the real-world settings part as Grgic et al. (2011) mention. In addition, the value of reproducible studies has to be further communicated by only relying on open-source software and data.

Appendix A

Attachments

A.1 Implementation

This chapter provides a detailed description of the implementation of the experiments. A short introduction explains the use of a parser and the setup for feature processing. This is followed by the implementation of the baseline, the rank list method, and the standardization method as described in section 4. Furthermore, it is described how the code profits from functional programming and, finally, how runtime is measured. The full code is publicly available on GitHub.[1]

A.1.1 Parser

For user-friendliness, a parser with predefined available choices is configured. The user can choose amongst all implemented comparison methods (for the rank list and the standardization method). The program uses predefined categories to easily evaluate whether the chosen one is a rank list or a standardization comparison method. Furthermore, when running standardization comparison methods, the user can choose between the preprocessing methods *standardize*, *subtract_mean*, or *omitted*. A specific protocol (i.e., *close*, *medium*, *far*) can also be selected. On-demand, a larger cohort is chosen (i.e., 86 instead of 43 samples), and calculation results such as the RRs or data files, including the computed scores used for FMR and FNMR, are recorded.

A.1.2 Setup

The setup includes the following steps to retrieve all data needed for the comparison methods. First, the features are extracted from all images in the SCface database using the ArcFace-Insight-Face-IResNet100 model and saved in the directory `samplewrapper-2/`. A database instance is created for a chosen protocol (i.e., *close*, *medium*, or *far*) from which all probe, gallery, and cohort samples are extracted. The development sets are used for probe and gallery samples and the training set for the cohort samples. The cohort is extended with the evaluation sets of the probe and the gallery images if a larger cohort is desired, as shown in listing A.1.

Next, the path to the extracted features is saved, and using Bob, the features are assigned to the corresponding samples. For this, the sample key is used, which is concatenated with the HDF5 file extension (see listing A.2).

[1]`https://github.com/maettuu/bob-face-recognition`

```python
import bob.bio.face
...
def extract_samples(protocol, enable_larger_cohort):
    # define database using chosen protocol and extract samples
    database = bob.bio.face.database.SCFaceDatabase(protocol)
    probes = database.probes()
    gallery = database.references()
    cohort = database.background_model_samples()

    if enable_bigger_cohort:
        cohort = cohort + \
                 database.references(group="eval") + \
                 database.probes(group="eval")

    return probes, gallery, cohort
```

Listing A.1: Samples Extraction

```python
import bob.io.base
import pathlib
...
# determine location of file
file_path = str(pathlib.Path().resolve())
directory_path = file_path + "/samples_pipe_all/samplewrapper-2/"
...
def load_features(sample):
    sample_key = sample.key
    # add the file extension to key
    new_sample_key = sample_key + ".h5"

    # load from destination
    sample_features = bob.io.base.load(directory_path + new_sample_key)
    sample.features = sample_features
```

Listing A.2: Assigning Features to Samples

After this step, sample sets are unwrapped in a simple for-loop such that the variables *probes*, *gallery*, and *cohort* only include separate samples. The setup is now complete, and the samples are ready to be used. Listing A.3 shows the definition of a sample object with the attributes important for the experiments (the attributes `rank_list` and `standardized_distances` are added later).

A.1.3 Baseline

As mentioned in section 4.1, the maximum score computed with $S_{cos}(u, v)$ (2.3) is most important as it represents the highest similarity. The computed similarity scores are first saved in a list. To report IP, the index of the maximum score is accessed using NumPy's `argmax` function. It reveals the chosen gallery image. If the subject IDs are the same, the match is positive and negative otherwise. For VP, the list of similarity scores for each probe and all gallery samples is saved in an external spreadsheet. Bob then takes this spreadsheet to plot ROC curves. Listing A.4 shows

```
Sample:
  key<string>                   features<numpyArray>
  capture<string>               rank_list<numpyArray>
  distance<string>              standardized_distances<numpyArray>
  subject_id<string>            ...
```

Listing A.3: Sample Object

the setup for the described baseline.

A.1.4 Rank List Method

First, the cohort is preprocessed for the rank list method, as mentioned in section 4.2. The `capture` attribute is accessed to split the cohort into the probe cohort and the gallery cohort. It either holds the string *mugshot*, which stands for high, or the string *surveillance*, which stands for low resolution. The `distance` attribute is accessed for each sample to filter the probe cohort with the defined protocol as shown in listing A.5. Since subject IDs are used as keys, the usage of dictionaries enables to easily sort the entries and guarantees equal order for probe and gallery samples being compared to the respective cohort.

As mentioned in section 4.2, the features for each subject are averaged, such that there is only one sample per subject in the probe cohort serving as a reference. To calculate the average of the features, NumPy's `mean` method is used on all the feature vectors which belong to the same subject (i.e., with the equal `subject_id` attribute). This process is shown in listing A.6.

Once all cosine distances are computed and saved in a list, as described in section 4.2, the rank list can be generated. This is done by converting the cosine distances using NumPy's `argsort` method as shown in listing A.7. It is applied twice, since the first time, the list is only converted into a list of pointers, where each value indicates at what index the corresponding ranked distance is saved. To generate a rank list, `argsort` is used a second time on this list of pointers.

Finally, listing A.8 uses all functions mentioned above and shows the entire preprocessing procedure of the cohort and the samples for the rank list method.

After this step, the probe and gallery samples are ready to be used for the different comparison methods (see section 2.4.2). The methods are applied using functional programming further described in section A.1.6 below.

```python
import numpy as np
import scipy.spatial
...
def baseline(probe_sample, gallery_samples):
    # instantiate list for calculated cosine distances
    cosine_distances = []

    for gallery_sample in gallery_samples:
        # calculate and save cosine distance between probe and current gallery sample
        cosine_distance = scipy.spatial.distance.cosine(
            probe_sample.features, gallery_sample.features
        )
        cosine_distances.append(cosine_distance)

        data = [probe_sample.reference_id, probe_sample.subject_id,
                gallery_sample.reference_id, gallery_sample.subject_id,
                -cosine_distance]
        # save to external spreadsheet to determine VP
        save_scores(data)

    # negate list to return cosine similarity
    return -np.array(cosine_distances)

...
def get_match_result(probe_sample, gallery_sample):
    # return 1 for positive matches (if the subject_ids are the same)
    if probe_sample.subject_id == gallery_sample.subject_id:
        return 1

    return 0
...
positive_matches = 0
...
for probe_sample in probe_samples:
    result = baseline(probe_sample, gallery_samples)
    # find maximum score and compare IDs for IP
    max_score_index = np.argmax(result)
    positive_matches += get_match_result(probe_sample, gallery_samples[max_score_index])
```

Listing A.4: Baseline Comparison

```python
def split_cohort(cohort_samples, protocol):
    # instantiate dictionaries for probe and gallery cohort samples
    cohort_probes = {}
    cohort_gallery = {}

    for sample in cohort_samples:
        # extract the 'capture' attribute
        curr_capture = sample.capture

        # check capture to distinguish between probe and gallery cohort
        if str(curr_capture) == 'surveillance':
            # check for the protocol used when defining the database
            if str(sample.distance) == protocol:
                curr_subject_id = sample.subject_id
                # add an entry if subject_id is not yet in the dictionary
                if curr_subject_id not in cohort_probes:
                    cohort_probes[curr_subject_id] = [sample.features]
                # else extract already recorded features and add current ones
                else:
                    cohort_probes.get(curr_subject_id).append(sample.features)
        elif str(curr_capture) == 'mugshot':
            cohort_gallery[sample.subject_id] = sample.features

    return cohort_probes, cohort_gallery
```

Listing A.5: Splitting and Filtering of Cohort

```python
import numpy as np
...
def calculate_average(cohort_probes):
    # instantiate dictionary for subjects w/ averaged features
    averaged_features = {}

    for subject_id, features in cohort_probes.items():
        # calculate the average of all features
        curr_average_features = np.mean(features, axis=0)
        # add average to the dictionary
        averaged_features[subject_id] = curr_average_features

    return averaged_features
```

Listing A.6: Compute Average of Features

```python
import scipy.spatial

def get_cosine_distances(sample, cohort_samples):
    # instantiate list for calculated cosine distances
    cosine_distances = []

    # sort keys to guarantee equal order
    for key in sorted(cohort_samples.keys()):
        # calculate and save cosine distance
        cosine_distances.append(
            scipy.spatial.distance.cosine(sample.features, cohort_samples[key])
        )

    return np.array(cosine_distances)


def generate_rank_list(samples, cohort_samples):
    for sample in samples:
        cosine_distances = get_cosine_distances(sample, cohort_samples)
        # use argsort to convert into an array of orders
        order = np.argsort(cosine_distances)
        # use argsort again to convert into a rank list and add it to sample
        sample.rank_list = np.argsort(order)
```

Listing A.7: Rank List Generation

```python
cohort_probes, cohort_gallery = split_cohort(cohort, protocol)
cohort_probes_averaged = calculate_average(cohort_probes)

# usage of rank lists -> generate rank lists
if category == "rank-list-comparison":
    generate_rank_list(probe_samples, cohort_probes_averaged)
    generate_rank_list(gallery_samples, cohort_gallery)
```

Listing A.8: Rank List Method Preprocessing

```python
def standardize(samples, cohort_samples):
    for sample in samples:
        cosine_distances = get_cosine_distances(sample, cohort_samples)
        # subtract the mean from the list and divide by the standard deviation
        sample.standardized_distances = np.divide(
            np.subtract(cosine_distances, np.mean(cosine_distances)),
            np.std(cosine_distances)
        )


def subtract_mean(samples, cohort_samples):
    for sample in samples:
        cosine_distances = get_cosine_distances(sample, cohort_samples)
        # subtract mean from list
        sample.standardized_distances = np.subtract(
            cosine_distances, np.mean(cosine_distances)
        )


def omitted(samples, cohort_samples):
    for sample in samples:
        cosine_distances = get_cosine_distances(sample, cohort_samples)
        # directly assign without standardization
        sample.standardized_distances = cosine_distances
```

Listing A.9: Standardized Lists

A.1.5 Standardization Method

Since the standardization method also uses the cohort, the steps shown in listings A.5 and A.6 are applied first. However, after computing the cosine distances, the lists are preprocessed depending on the user's choice as shown in listing A.9. Either the methods *standardize* (4.1) or *subtract_mean* (4.2) are applied, or the step is *omitted* altogether. The entire preprocessing procedure is described in listing A.10.

The following section explains how Python's `eval` method is used to call the preprocessing method chosen by the user. After preprocessing, the probe and gallery samples are ready to be used for the distance computations listed in section 3.4. Again functional programming is applied to select the functions' computations.

```python
cohort_probes, cohort_gallery = split_cohort(cohort, protocol)
cohort_probes_averaged = calculate_average(cohort_probes)
...
# usage of lists w/o converting to rank -> standardize lists
elif category == "standardization_comparison":
    standardization_function = eval(standardization_method)
    standardization_function(probe_samples, cohort_probes_averaged)
    standardization_function(gallery_samples, cohort_gallery)
```

Listing A.10: Standardization Method Preprocessing

A.1.6 Dynamic Execution

By default, the program runs all comparison methods for all protocols. To follow the principles of good code structure, duplicated lines of code are prevented. This is achieved using Python's `eval` method on the specified comparison method. In general, this Python method poses some security risks. However, since the comparison methods available are given as choices for the parser, the user is not able to input anything other than what is predefined, therefore guaranteeing a successful execution. Since each comparison method has a corresponding function, the two are dynamically linked without additional code. This approach is applied equally to the user's choice of preprocessing for the standardization method. Listing A.11 shows an example for the similarity function $S_{2010mueller}(\pi, \gamma)$ (2.6). Note that the normalization factor is removed since only the highest score is necessary (i.e., the highest score remains highest after normalization). The code has a function just like this for every comparison method. Again, the positive matches are recorded for IP, and all score results are saved in an external spreadsheet used by Bob for VP.

A.1.7 Runtime

Execution time is measured running the terminal only, with no running background tasks. For this, an Intel(R) Core(TM) i7-8650U CPU at a rate of 1.90GHz is used.

Measurements are taken using the `process_time()` function provided by Python's `time` module[2]. The first measurement includes the time for preprocessing (i.e., the time it takes for generating the rank and standardized lists). It is measured separately because the baseline does not have to preprocess and use the cohort. The second measurement is started after the preprocessing steps described in sections A.1.2-A.1.5 before the program starts computing similarity scores for each probe and all gallery samples. It is stopped after the last comparison before computing the RR. This guarantees the measurement to include the comparison process only.

It follows that for the rank list and the standardization method, the two measurements are added to determine the total runtime. In contrast, for the baseline, only the second measurement is relevant. By default, all protocols are executed for one comparison method, so those runtime measurements are averaged. However, if a specific protocol is chosen, the corresponding runtime is returned directly.

[2] `https://docs.python.org/3/library/time.html`, retrieved 14 July, 2022

```python
import math
comparison_method = "mueller2010"
...
def mueller2010(probe_sample, gallery_sample):
    # initialize score
    similarity_score = 0
    # create a list of tuples including both ranks at the same index
    rank_tuples = zip(probe_sample.rank_list, gallery_sample.rank_list)
    # loop through rank lists and compute similarity
    for probe_rank, gallery_rank in rank_tuples:
        similarity_score += 1 / math.sqrt(probe_rank + gallery_rank + 1)

    return similarity_score

...
def get_similarity_scores(probe_sample, gallery_samples, comparison_function):
    # instantiate list for calculated similarity scores between
    # probe sample rank list and all gallery sample rank lists
    similarity_scores = []

    for gallery_sample in gallery_samples:
        similarity_score = comparison_function(probe_sample, gallery_sample)
        # append the score to the list
        similarity_scores.append(similarity_score)

        data = [probe_sample.reference_id, probe_sample.subject_id,
                gallery_sample.reference_id, gallery_sample.subject_id,
                similarity_score]
        # save to external spreadsheet to determine VP
        save_scores(data)

    return np.array(similarity_scores)

...
comparison_function = eval(comparison_method)
...
for probe_sample in probe_samples:
    result = get_similarity_scores(probe_sample, gallery_samples, comparison_function)
    # find maximum score and compare IDs for IP
    max_score_index = np.argmax(result)
    positive_matches += get_match_result(probe_sample, gallery_samples[max_score_index])
```

Listing A.11: Dynamic Comparison Method

A.2 Additional Results for Different Parameters

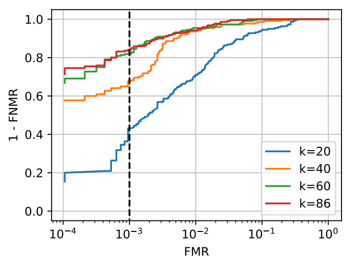
A.2.1 Schroff's, Müller's & Wartmann's Parameters

Schroff

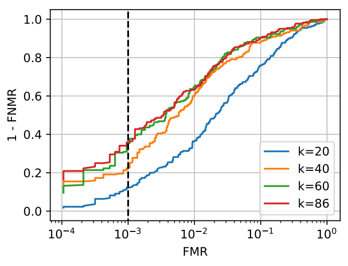
For the large cohort, which includes 86 samples, values $k \in \{20, 40, 60, 86\}$ are presented to show a gradual increase. Table A.1 summarizes IP. Figure A.1 shows VP for each protocol.

parameter	RR (%)		
	close	*medium*	*far*
$k = 20$	73.18	44.09	14.55
$k = 40$	91.82	65.45	19.09
$k = 60$	96.36	**71.82**	23.64
$k = 86$	**97.28**	71.36	**27.27**

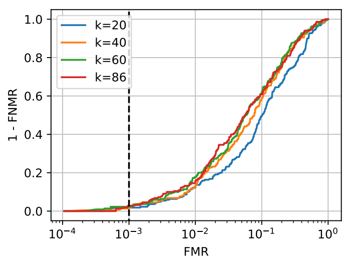
Table A.1: SCHROFF IP LARGE COHORT. This table shows the RRs for each defined k as well as for each protocol given in percentages using the large cohort. The best values for each protocol are marked in **bold**.



(a) *close*



(b) *medium*



(c) *far*

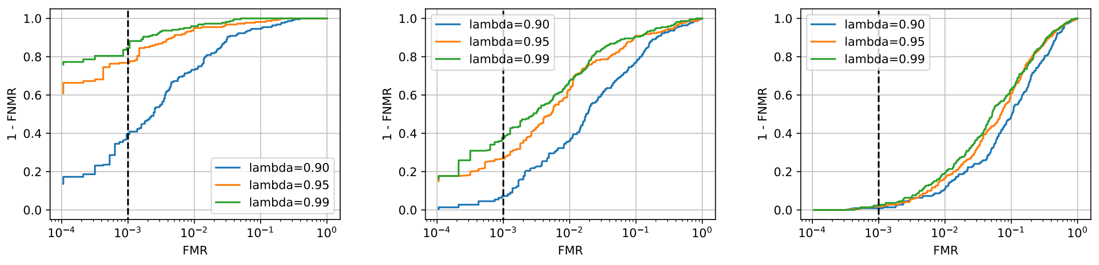
Figure A.1: SCHROFF VP LARGE COHORT. This figure including subfigures (a) - (c) shows VP of the similarity function $S_{schroff}$ (2.7) for each protocol on the large cohort.

Müller 2013

For the large cohort, the values $\lambda \in \{0.90, 0.95, 0.99\}$ are presented in table A.2 and figure A.2 to show a gradual increase.

parameter	RR (%)		
	close	*medium*	*far*
$\lambda = 0.90$	76.82	45.91	16.82
$\lambda = 0.95$	92.27	65.45	20.91
$\lambda = 0.99$	**97.27**	**73.64**	**27.27**

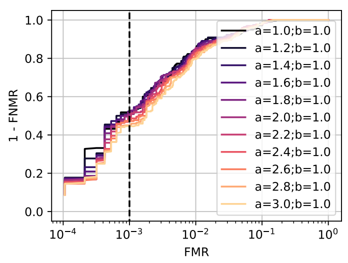
Table A.2: MÜLLER 2013 IP LARGE COHORT. This table shows the RRs for each defined λ as well as for each protocol given in percentages using the large cohort. The best values for each protocol are marked in **bold**.



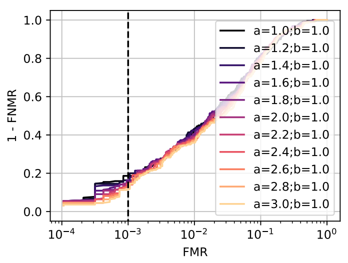
(a) *close* (b) *medium* (c) *far*

Figure A.2: MÜLLER 2013 VP LARGE COHORT. This figure including subfigures (a) - (c) shows VP of the similarity function $S_{2013mueller}$ (2.8) for each protocol using a different parameter λ on the large cohort.

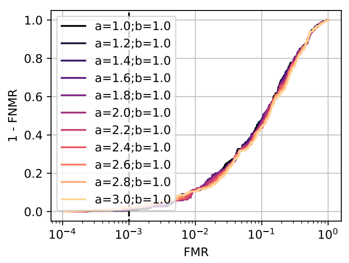
Wartmann



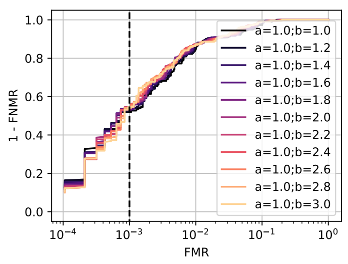
(a) increasing α, *close*



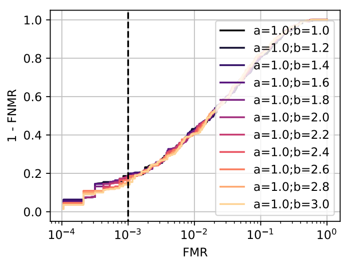
(b) increasing α, *medium*



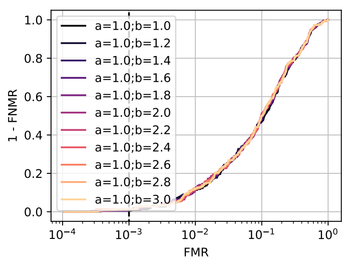
(c) increasing α, *far*



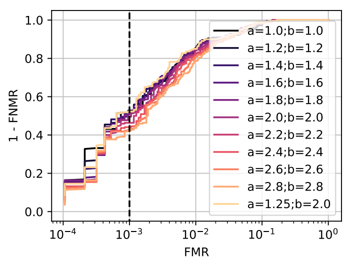
(d) increasing β, *close*



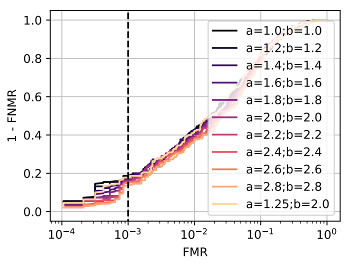
(e) increasing β, *medium*



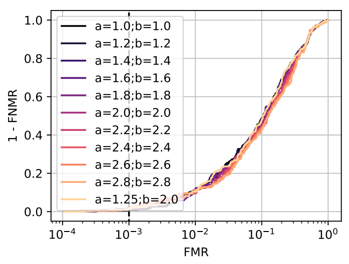
(f) increasing β, *far*



(g) increasing both parameters, *close*

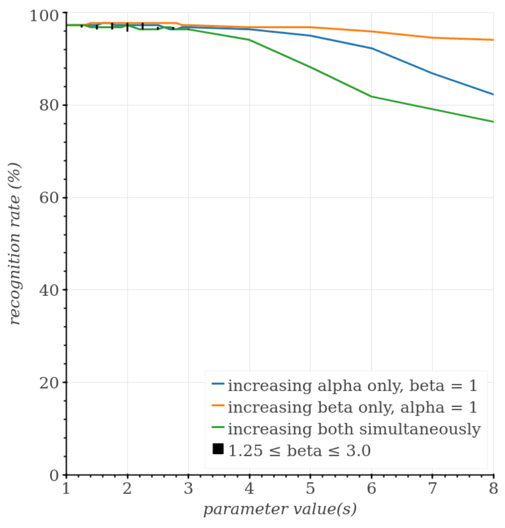


(h) increasing both parameters, *medium*

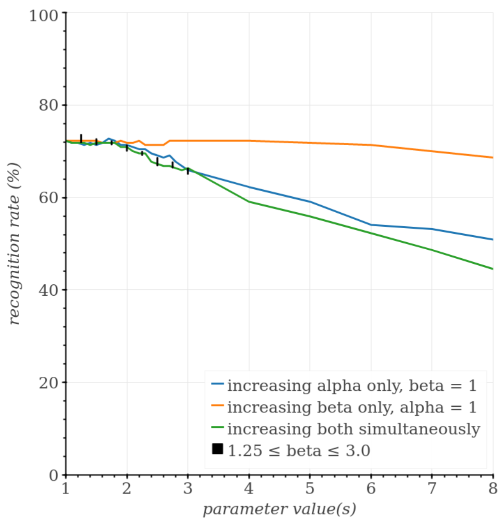


(i) increasing both parameters, *far*

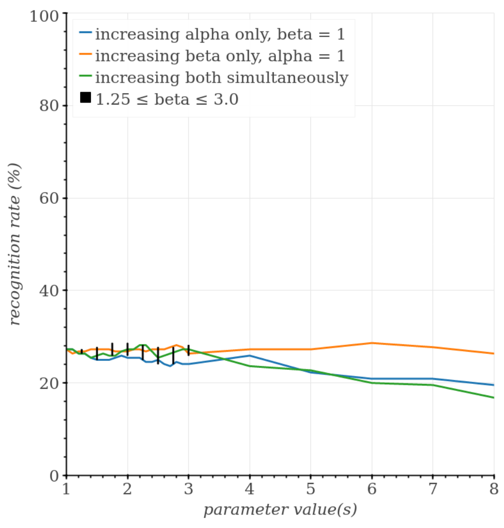
Figure A.3: ADDITIONAL WARTMANN VP SMALL COHORT. This figure including subfigures (a) - (i) shows VP of the similarity function $S_{wartmann}$ (2.10) for various α-β-pairs. When increasing both parameters simultaneously, solely $\alpha = 1.25$ and $\beta = 2.0$ shows a considerable difference, however the performance is practically equal to that of $\alpha = 1.0$ and $\beta = 1.0$.



(a) *close*



(b) *medium*



(c) *far*

Figure A.4: WARTMANN IP LARGE COHORT. This figure including subfigures (a) - (c) shows IP for various α-β-pairs on the large cohort. The black quads show the range of resulting RRs where α equals the value on the x-axis and $\beta \in [1.25, 3.0]$.

parameter	RR (%) *close*	*medium*	*far*
$\alpha = 1.0, \beta = 1.0$	97.27	72.27	**27.27**
$\alpha = 1.1, \beta = 1.0$	97.27	71.82	**27.27**
$\alpha = 1.2, \beta = 1.0$	97.27	71.82	26.36
$\alpha = 1.3, \beta = 1.0$	97.27	71.36	26.36
$\alpha = 1.4, \beta = 1.0$	97.27	71.82	25.45
$\alpha = 1.5, \beta = 1.0$	97.27	71.36	25.00
$\alpha = 1.6, \beta = 1.0$	**97.73**	71.82	25.00
$\alpha = 1.7, \beta = 1.0$	**97.73**	**72.73**	25.00
$\alpha = 1.8, \beta = 1.0$	97.27	72.27	25.45
$\alpha = 1.9, \beta = 1.0$	97.27	71.36	25.91
$\alpha = 2.0, \beta = 1.0$	97.27	71.36	25.45

(a) increasing α

parameter	RR (%) *close*	*medium*	*far*
$\alpha = 1.0, \beta = 1.1$	97.27	**72.27**	26.36
$\alpha = 1.0, \beta = 1.2$	97.27	**72.27**	26.82
$\alpha = 1.0, \beta = 1.3$	97.27	**72.27**	26.82
$\alpha = 1.0, \beta = 1.4$	**97.73**	**72.27**	**27.27**
$\alpha = 1.0, \beta = 1.5$	**97.73**	**72.27**	**27.27**
$\alpha = 1.0, \beta = 1.6$	**97.73**	71.82	**27.27**
$\alpha = 1.0, \beta = 1.7$	**97.73**	71.82	**27.27**
$\alpha = 1.0, \beta = 1.8$	**97.73**	71.82	26.82
$\alpha = 1.0, \beta = 1.9$	**97.73**	**72.27**	26.82
$\alpha = 1.0, \beta = 2.0$	**97.73**	71.82	26.82
$\alpha = 1.0, \beta = 2.1$	**97.73**	71.82	**27.27**

(b) increasing β

parameter	RR (%) *close*	*medium*	*far*
$\alpha = 1.2, \beta = 1.2$	**97.27**	**71.82**	26.36
$\alpha = 1.3, \beta = 1.3$	**97.27**	**71.82**	26.36
$\alpha = 1.4, \beta = 1.4$	96.82	71.36	25.45
$\alpha = 1.5, \beta = 1.5$	96.82	**71.82**	25.91
$\alpha = 1.6, \beta = 1.6$	96.82	**71.82**	26.36
$\alpha = 1.7, \beta = 1.7$	96.82	**71.82**	25.91
$\alpha = 1.8, \beta = 1.8$	96.82	**71.82**	25.91
$\alpha = 1.9, \beta = 1.9$	96.82	70.91	26.82
$\alpha = 2.0, \beta = 2.0$	**97.27**	70.91	27.27
$\alpha = 2.1, \beta = 2.1$	96.82	70.00	27.27
$\alpha = 2.2, \beta = 2.2$	96.36	69.55	**28.18**

(c) increasing both

parameter	RR (%) *close*	*medium*	*far*
$\alpha = 1.25, \beta = 2.50$	**97.27**	72.73	26.82
$\alpha = 1.25, \beta = 2.75$	96.82	**73.64**	27.27
$\alpha = 1.25, \beta = 3.00$	96.82	72.73	26.82
$\alpha = 1.75, \beta = 2.25$	96.36	71.82	27.73
$\alpha = 1.75, \beta = 2.50$	96.82	72.27	**28.64**
$\alpha = 1.75, \beta = 2.75$	96.36	71.82	28.18
$\alpha = 1.75, \beta = 3.00$	96.36	71.82	27.27
$\alpha = 2.00, \beta = 1.75$	**97.27**	70.91	26.36
$\alpha = 2.00, \beta = 2.25$	96.36	70.45	28.18
$\alpha = 2.00, \beta = 2.50$	95.91	70.45	28.18
$\alpha = 2.00, \beta = 2.75$	96.36	70.00	**28.64**

(d) various α-β-pairs

Table A.3: WARTMANN IP LARGE COHORT. This table including subtables (a) - (d) shows the RRs for each defined α-β-pair as well as for each protocol given in percentages. The best values for each protocol are marked in **bold**.

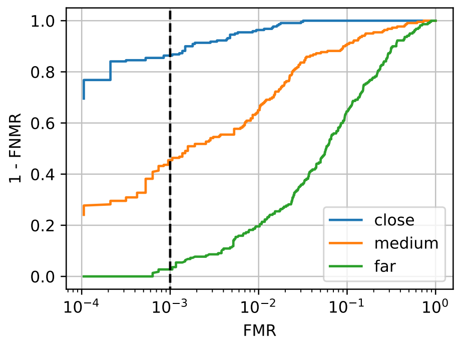
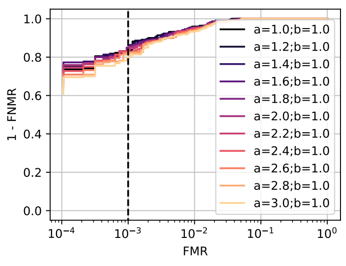
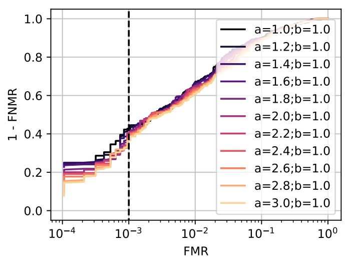
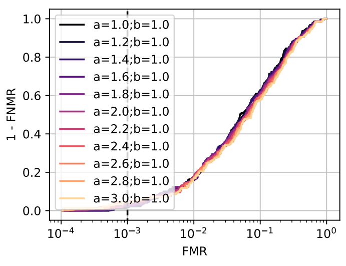
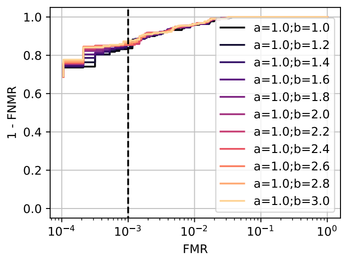


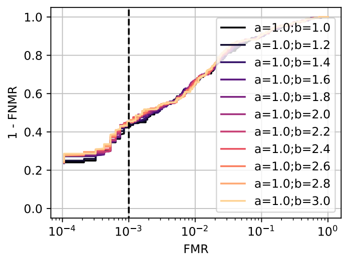
Figure A.5: WARTMANN VP LARGE COHORT. This figure shows VP of the similarity function $S_{wartmann}$ (2.10). Here $\alpha = 1.0$ and $\beta = 2.8$ for best performance; however, as figure A.6 shows, the values have a small impact on overall performance.

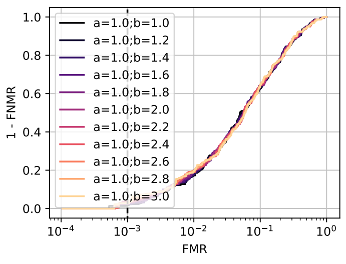

(a) increasing α *close*

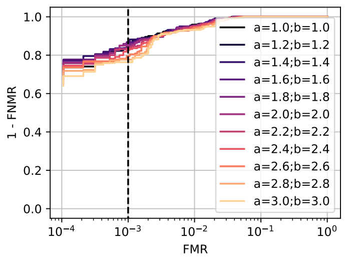

(b) increasing α *medium*

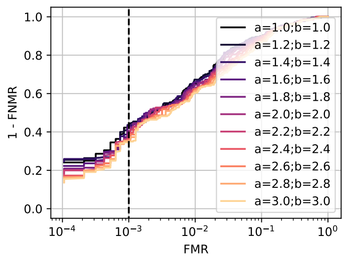

(c) increasing α *far*


(d) increasing β *close*


(e) increasing β *medium*


(f) increasing β *far*


(g) increasing both parameters *close*


(h) increasing both parameters *medium*

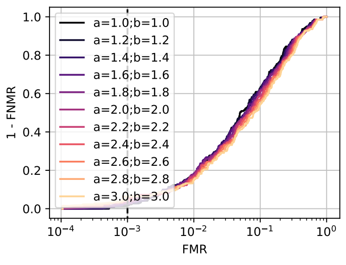
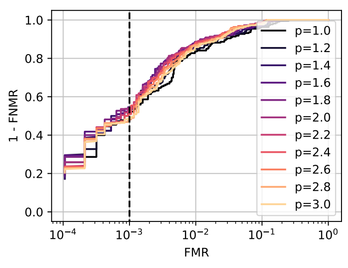
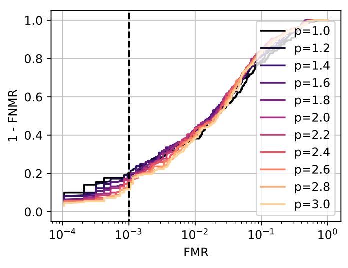

(i) increasing both parameters *far*

Figure A.6: ADDITIONAL WARTMANN VP LARGE COHORT. This figure including subfigures (a) - (i) shows VP of the similarity function $S_{wartmann}$ (2.10) for various α-β-pairs on the large cohort.

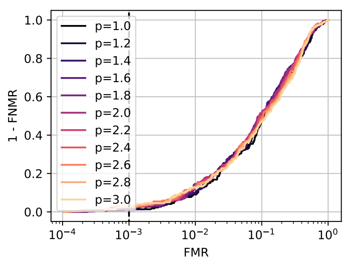
A.2.2 Minkowski's Parameter



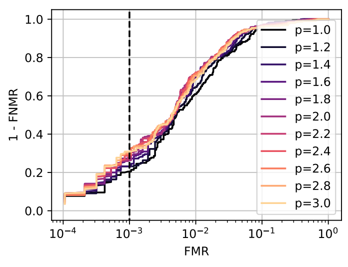
(a) *standardize, close*



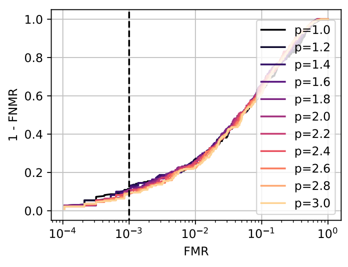
(b) *standardize, medium*



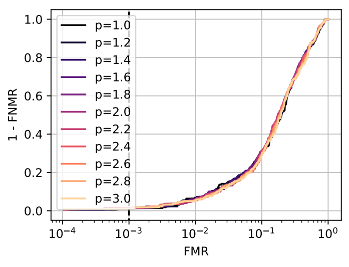
(c) *standardize, far*



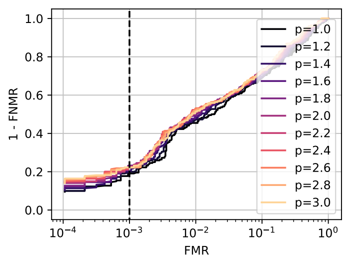
(d) *subtract_mean, close*



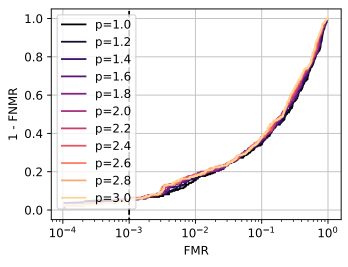
(e) *subtract_mean, medium*



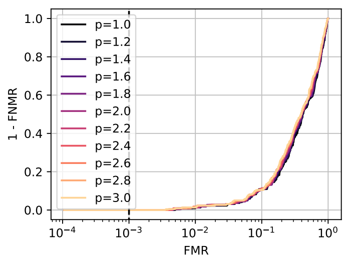
(f) *subtract_mean, far*



(g) *omitted, close*

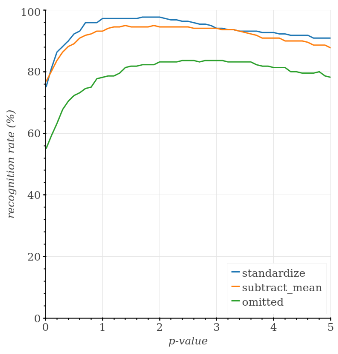


(h) *omitted, medium*

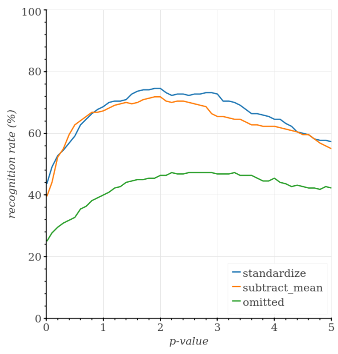


(i) *omitted, far*

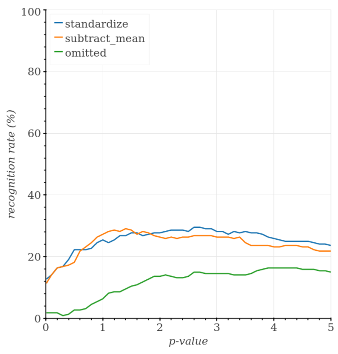
Figure A.7: ADDITIONAL MINKOWSKI VP SMALL COHORT. This figure including subfigures (a) - (i) shows VP of the similarity function $S_{minkowski}$ (3.5) for various p-values and preprocessing methods on the small cohort.



(a) *close*

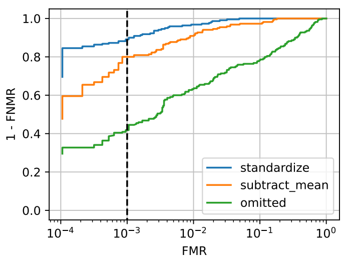


(b) *medium*

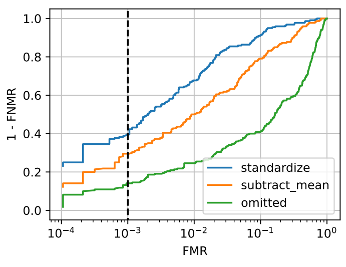


(c) *far*

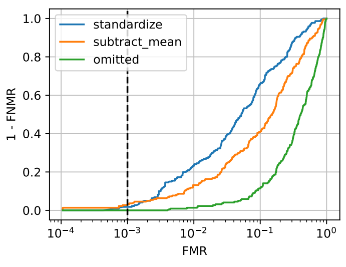
Figure A.8: MINKOWSKI IP LARGE COHORT. This figure including subfigures (a) - (c) shows IP for various p-values on the large cohort.



(a) *close*



(b) *medium*



(c) *far*

Figure A.9: MINKOWSKI VP LARGE COHORT. This figure including subfigures (a) - (c) shows VP of the similarity function $S_{minkowski}$ (3.5). Here $p = 2.0$ for best performance (i.e., same VP as $S_{sq.euclidean}$ (3.4)), however as figure A.10 shows, the value has a small impact.

parameter	RR (%)		
	close	*medium*	*far*
$p = 2.0$	**97.73**	**74.55**	27.73
$p = 2.1$	97.27	73.18	28.18
$p = 2.2$	96.82	72.27	28.64
$p = 2.3$	96.82	72.73	28.64
$p = 2.4$	96.36	72.73	28.64
$p = 2.5$	96.36	72.27	28.18
$p = 2.6$	95.91	72.73	**29.55**
$p = 2.7$	95.45	72.73	**29.55**
$p = 2.8$	95.45	73.18	29.09
$p = 2.9$	95.00	73.18	29.09
$p = 3.0$	94.09	72.73	28.18

(a) *standardize*

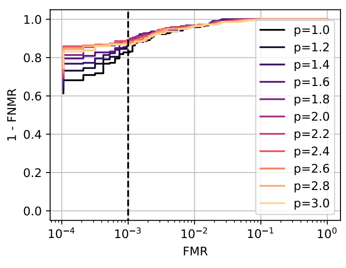
parameter	RR (%)		
	close	*medium*	*far*
$p = 1.2$	94.55	69.09	28.64
$p = 1.3$	94.55	69.55	28.18
$p = 1.4$	**95.00**	70.00	**29.09**
$p = 1.5$	94.55	69.55	28.64
$p = 1.6$	94.55	70.00	27.27
$p = 1.7$	94.55	70.91	28.18
$p = 1.8$	94.55	71.36	27.73
$p = 1.9$	**95.00**	**71.82**	26.82
$p = 2.0$	94.55	**71.82**	26.36
$p = 2.1$	94.55	70.45	25.91
$p = 2.2$	94.55	70.00	26.36

(b) *subtract_mean*

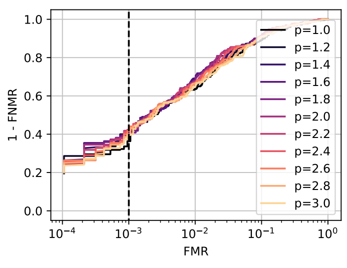
parameter	RR (%)		
	close	*medium*	*far*
$p = 2.2$	83.18	**47.27**	13.64
$p = 2.3$	83.18	46.82	13.18
$p = 2.4$	**83.64**	46.82	13.18
$p = 2.5$	**83.64**	**47.27**	13.64
$p = 2.6$	**83.64**	**47.27**	15.00
$p = 2.7$	83.18	**47.27**	15.00
$p = 2.8$	**83.64**	**47.27**	14.55
$p = 2.9$	**83.64**	**47.27**	14.55
$p = 3.0$	**83.64**	46.82	14.55
$p = 3.1$	**83.64**	46.82	14.55
$p = 3.2$	83.18	46.82	14.55
$p = 4.2$	81.36	43.64	**16.36**

(c) *omitted*

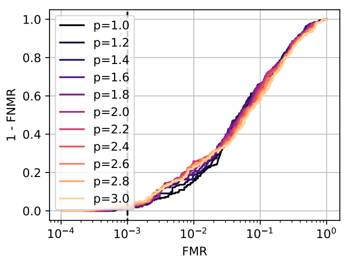
Table A.4: MINKOWSKI IP LARGE COHORT. This table including subtables (a) - (c) shows the RRs for each defined p as well as for each protocol given in percentages. The best values for each protocol are marked in **bold**.



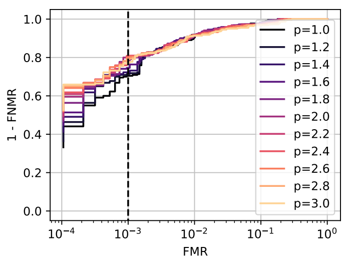
(a) *standardize, close*



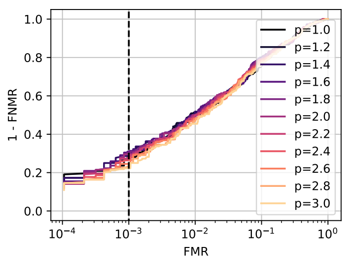
(b) *standardize, medium*



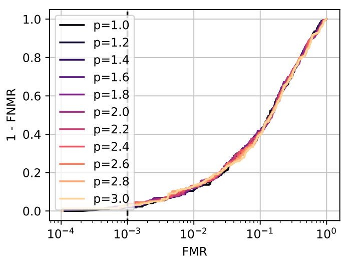
(c) *standardize, far*



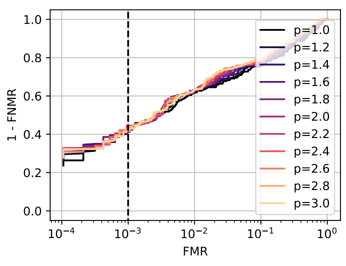
(d) *subtract_mean, close*



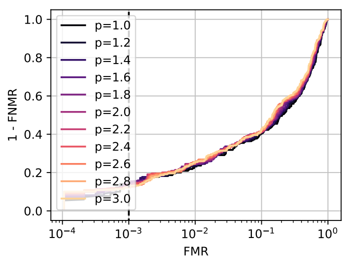
(e) *subtract_mean, medium*



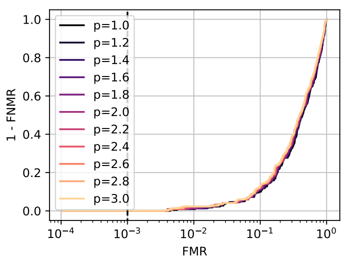
(f) *subtract_mean, far*



(g) *omitted, close*



(h) *omitted, medium*



(i) *omitted, far*

Figure A.10: ADDITIONAL MINKOWSKI VP LARGE COHORT. This figure including subfigures (a) - (i) shows VP of the similarity function $S_{minkowski}$ (3.5) for various p-values and preprocessing methods on the large cohort.

List of Figures

List of Tables

List of Listings

Acronyms

CNN	Convolutional Neural Network. 6, 14, 44, 45
CPU	Central Processing Unit. 23, 54
EER	Equal Error Rate. 5, 6
FAR	False Acceptance Rate. 5
FMR	False Match Rate. 5, 47
FNMR	False Non-Match Rate. 5, 47
FNR	False Negative Rate. 5
FPR	False Positive Rate. 5
FRR	False Rejection Rate. 5
HE	Histogram Equalization. 4, 6
IP	Identification Performance. 5, 10, 19, 23–30, 32–35, 37, 38, 42–44, 48, 54, 56, 57, 59, 60, 63, 64, 66, 67
ISV	Inter-Session Variability. 5
LBP	Local Binary Patterns. 4–6
LDA	Linear Discrimination Analysis. 1, 5
LDA-IR	LDA-Infrared. 5
LGBPHS	Local Gabor Binary Pattern Histogram Sequence. 1, 5, 6
LRPCA	Local Region PCA. 5
PCA	Principal Component Analysis. 1, 5, 6
ROC	Receiver Operating Characteristics. 5, 6, 23, 28, 30, 32, 34, 37, 42, 48
RR	Recognition Rate. 5, 23–30, 33–35, 37, 38, 42, 43, 47, 54, 56, 57, 59, 60, 64
SQ	Self Quotient. 4
T&T	Tan & Trigg's. 4
TMR	True Match Rate. 5
VP	Verification Performance. 5, 6, 10, 16, 19, 23–26, 28–32, 34, 36, 37, 39, 42–44, 48, 54, 56–58, 61–63, 65, 66

Bibliography

Anjos, A., El-Shafey, L., Wallace, R., Günther, M., McCool, C., and Marcel, S. (2012). Bob: A free signal processing and machine learning toolbox for researchers. In *Proceedings of the 20th ACM International Conference on Multimedia*, page 1449–1452. Association for Computing Machinery.

Anjos, A., Günther, M., de Freitas Pereira, T., Korshunov, P., Mohammadi, A., and Marcel, S. (2017). Continuously reproducing toolchains in pattern recognition and machine learning experiments. In *International Conference on Machine Learning*.

Cao, Q., Shen, L., Xie, W., Parkhi, O. M., and Zisserman, A. (2018). VGGFace2: A dataset for recognising faces across pose and age. In *IEEE International Conference on Automatic Face & Gesture Recognition*, pages 67–74.

de Freitas Pereira, T., Schmidli, D., Linghu, Y., Zhang, X., Marcel, S., and Günther, M. (2022). Eight years of face recognition research: Reproducibility, achievements and open issues. In *IEEE Conference on Computer Vision and Pattern Recognition*. arXiv.

de Leeuw, K. and Bergstra, J. (2007). *The History of Information Security: A Comprehensive Handbook*. Elsevier Science.

Deng, J., Guo, J., Xue, N., and Zafeiriou, S. (2019). ArcFace: Additive angular margin loss for deep face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4690–4699.

Grgic, M., Delac, K., and Grgic, S. (2011). SCface – surveillance cameras face database. *Multimed Tools Appl*, 51:863–879.

Guo, G. and Zhang, N. (2019). A survey on deep learning based face recognition. *Computer Vision and Image Understanding*, 189:13–49.

Guresen, E. and Kayakutlu, G. (2011). Definition of artificial neural networks with comparison to other networks. *Procedia Computer Science*, 3:426–433.

Günther, M., Haufe, D., and Würtz, R. P. (2012a). Face recognition with disparity corrected gabor phase differences. In *International Conference on Artificial Neural Networks and Machine Learning*, pages 411–418. Springer Berlin Heidelberg.

Günther, M., Shafey, L. E., and Marcel, S. (2016). Face recognition in challenging environments: An experimental and reproducible research survey. In *Face Recognition Across the Imaging Spectrum*, page 247–280. Springer International Publishing, Cham.

Günther, M., Shafey, L. E., and Marcel, S. (2017). 2D face recognition: An experimental and reproducible research survey. Technical Report Idiap-RR-13-2017, Idiap.

Günther, M., Wallace, R., and Marcel, S. (2012b). An open source framework for standardized comparisons of face recognition algorithms. In *European Conference on Computer Vision*, pages 547–556. Springer Berlin Heidelberg.

Heusch, G., Rodriguez, Y., and Marcel, S. (2006). Local binary patterns as an image preprocessing for face authentication. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 9–14.

Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105.

Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., and Song, L. (2017). SphereFace: Deep hypersphere embedding for face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 212–220.

Liu, W., Wen, Y., Yu, Z., and Yang, M. (2016). Large-margin softmax loss for convolutional neural networks. *IEEE Conference on Machine Learning*, 48(3):507–516.

Lui, Y. M., Bolme, D., Phillips, P. J., Beveridge, J. R., and Draper, B. A. (2012). Preliminary studies on the good, the bad, and the ugly face recognition challenge problem. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 9–16.

Masi, I., Wu, Y., Hassner, T., and Natarajan, P. (2018). Deep face recognition: A survey. In *IEEE Conference on Graphics, Patterns and Images*, pages 471–478.

Metz, C. E. (1978). Basic principles of roc analysis. *Seminars in Nuclear Medicine*, 8(4):283–298.

Mohamad Mezher, A., Iturria Rivera, P. E., Dueñas Santos, C. L., Cárdenas-Barrera, J., Meng, J., and Castillo Guerra, E. (2019). Minkowski distance order effect on the optimization of key devices positions in large-scale RF mesh networks. In *ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, pages 73–77.

Müller, M. K. (2010). *Lernen von Identitätserkennung unter Bildvariation*. Doctoral thesis, Ruhr-Universität Bochum.

Müller, M. K., Heinrichs, A., Tewes, A. H. J., Schäfer, A., and Würtz, R. P. (2007). Similarity rank correlation for face recognition under unenrolled pose. In *Advances in Biometrics*, pages 67–76. Springer Berlin Heidelberg.

Müller, M. K., Tremer, M., Bodenstein, C., and Würtz, R. P. (2013). Learning invariant face recognition from examples. *Neural Networks*, 41:137–146. Special Issue on Autonomous Learning.

O'Toole, A. J., Phillips, P. J., Jiang, F., Ayyad, J., Penard, N., and Abdi, H. (2007). Face recognition algorithms surpass humans matching faces over changes in illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(9):1642–1646.

Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2019). Deep face recognition. In *British Machine Vision Conference*, pages 1–12.

Phillips, P. J., Beveridge, J. R., Draper, B. A., Givens, G., O'Toole, A. J., Bolme, D. S., Dunlop, J., Lui, Y. M., Sahibzada, H., and Weimer, S. (2011). An introduction to the good, the bad, and the ugly face recognition challenge problem. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 346–353.

Ramírez-Gutiérrez, K., Cruz-Pérez, D., and Pérez-Meana, H. (2010). Face recognition and verification using histogram equalization. In *Applied Computer Science*, pages 85–89. WSEAS.

Schmidli, D. (2021). *Face Recognition Aspects with DNNs: An Experimental and Reproducible Research Survey*. Bachelor thesis, Universität Zürich.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823.

Schroff, F., Treibitz, T., Kriegman, D., and Belongie, S. (2011). Pose, illumination and expression invariant pairwise face-similarity measure via doppelgänger list comparison. In *IEEE International Conference on Computer Vision*, pages 2494–2501.

Shaha, P., Sharma, U., and Pawar, K. (2008). Face recognition technology. *International Journal of Research in Engineering, Science and Management*, 1(9):149–151.

Sirovich, L. and Kirby, M. (1987). Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A*, 4(3):519–524.

Spearman, C. (1987). The proof and measurement of association between two things. *The American Journal of Psychology*, 100(3/4):441–471.

Tan, X. and Triggs, B. (2010). Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE Transactions on Image Processing*, 19(6):1635–1650.

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

Vigna, S. (2015). A weighted correlation index for rankings with ties. In *Proceedings of the 24th International Conference on World Wide Web*, page 1166–1176. International World Wide Web Conferences Steering Committee.

Wallace, R., McLaren, M., McCool, C., and Marcel, S. (2011). Inter-session variability modelling and joint factor analysis for face authentication. In *IEEE International Joint Conference on Biometrics*, pages 1–8.

Wang, H., Li, S. Z., and Wang, Y. (2004). Face recognition under varying lighting conditions using self quotient image. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 819–824.

Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., Li, Z., and Liu, W. (2018). CosFace: Large margin cosine loss for deep face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5265–5274.

Wang, M. and Deng, W. (2021). Deep face recognition: A survey. *Neurocomputing*, 429:215–244.

Wartmann, T. M. (2021). *Frontal to Profile Face Recognition with Rank Lists*. Bachelor thesis, Universität Zürich.

Wen, Y., Zhang, K., Li, Z., and Qiao, Y. (2016). A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515. Springer International Publishing.

Yi, D., Lei, Z., Liao, S., and Li, S. Z. (2014). Learning face representation from scratch. In *IEEE Conference on Computer Vision and Pattern Recognition*. arXiv.

Zhang, W., Shan, S., Gao, W., Chen, X., and Zhang, H. (2005). Local gabor binary pattern histogram sequence (LGBPHS): A novel non-statistical model for face representation and recognition. In *IEEE International Conference on Computer Vision*, volume 1, pages 786–791.

Zhao, W., Chellappa, R., Phillips, P. J., and Rosenfeld, A. (2003). Face recognition: A literature survey. *ACM Comput. Surv.*, 35(4):399–458.

Zhao, W., Krishnaswamy, A., Chellappa, R., Swets, D. L., and Weng, J. (1998). *Discriminant Analysis of Principal Components for Face Recognition*, pages 73–85. Springer Berlin Heidelberg.