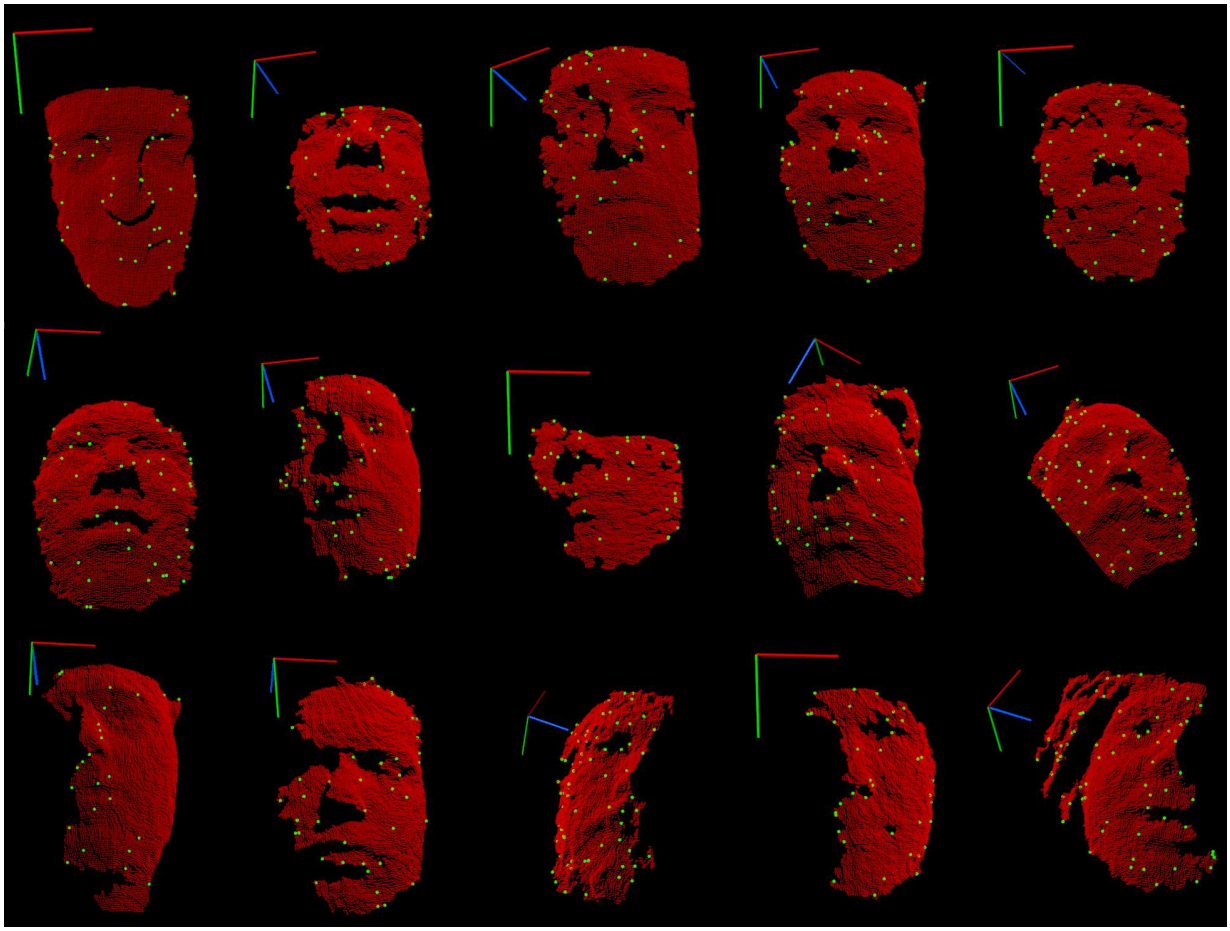
Machine Learning for 3D Face Recognition using off-the-shelf sensors

Florin Schimbinschi
s1982486
Department of Artificial Intelligence, University of Groningen

31 August 2013



Supervisors:

Dr. Marco Wiering *Artificial Intelligence, University of Groningen*
Prof. Lambert Schomaker *Artificial Intelligence, University of Groningen*

university of groningen / faculty of mathematics and natural sciences / artificial intelligence

Contents

Dintr-alte țări, de soare pline,
Pe unde-ați fost și voi străine,
Veniți, dragi păsări, înapoi
Veniți cu bine!

In memory of my grandparents

Acknowledgments

My most sincere gratitude and esteem to my supervisors Dr. Marco Wiering and Prof. Dr. Lambert Schomaker for their patience, guidance and most constructive practical and philosophical discussions during my research endeavours.

I am thankful to my family for believing in me, their outright support and for making it possible for me to follow my dreams and ambitions.

I would also like to thank Vali Codreanu for providing access to the 12 core machine of the visualization lab, that considerably reduced time while running experiments.

Abstract

The human brain is inherently hardwired to read psychological state before identity, hence its robustness towards the dynamic nature of faces and viewpoint changes. The novelty of this research consists in learning abstract atomical representations of shape cues, thus having the potential to solve multiple classification problems since a depiction unit can have multiple task-specific attributes. Machine learning versatility is paramount and as such multiple ensembles of varying complexity are trained based on unsupervised specialization of experts. A dataset of 18 individuals was recorded based on different variances in pose, expression and distance to the Kinect sensor. Using 3D object feature descriptors, the performance for face recognition is studied over 36 variance-specific pair tests, concluding that simple ensembles outperform complex ones, since the utility of each expert is highly dependent on the sampling resolution, distance metric and type of features. The methodology is robust towards occlusions and the performance can reach accuracies up to 90% depending on the complexity of the dataset, despite that there is no human supervision for generating the face region labels.

Chapter 1

Introduction

Our brains are prediction machines and we are starting to realize that we are reaching their limitations. Human existence is emergent, nature has done its part, it is now time to acknowledge that to better understand ourselves and the world we need to create a form of omniscience that can integrate information more efficiently for us. The faster we can process information, the better our predictions.

With the rise of the internet, we can observe an increasing trend in the amount of data being collected. In some sense, the internet can be considered as a form of distributed intelligence. Whether consciously or not, we are the creators of this emergent organism. We are feeding the machine what it was designed to eat: data. In return, we receive personalized, integrated information. This is one of the many goals of Artificial Intelligence (AI).

The synergy between man and machine becomes more evident if we consider that this organism can not accurately perceive nor interact with the environment directly — it does this through us. If we think of humankind as one organism and the internet as an artificial learning organism then we are witnessing a symbiotic relationship between the two. The logical conclusion is then to make use of this nascent organism to integrate information faster for us. We can observe this phenomenon in websites which collect data to perform user behavior mining. Everything is becoming personalized: social websites, e-mail, news, etc. Computers are far better than humans at performing fast computations yet they lack the basic means of interacting with the world.

While personalization has become a completely ubiquitous keyword on the internet, perhaps as a consequence, human conscience has simultaneously immersed itself even more into the digital world. The current research aims to make a step towards taking personalization out of the digital world into the real life, hopefully paving the way for intelligent environments. By extending the awareness of machines into the material world we can shift our behaviors and perceptions from being dependent on windows into the binary realm.

Giving machines the ability to read identity is the first step towards personalizing and augmenting our environments. This has the potential to change the way we connect and interact with information. Having aggregated data readily available for us, the AI could for example aid us in organizational tasks by providing the relevant information for us based on our preferences, the time of the day and other circumstances.

Face perception is perhaps the most highly developed visual skill in humans. The evolution of our perceptual systems has taken a very long time to reach current capabilities. However, machines do not necessarily have to possess the same type of sensors as we do. We have the power to control the type of

sensory information that machines will have. Since it is not feasible or mandatory to "evolve" similar sensory modules as those of humans, this research makes use of off-the-shelf infrared depth sensors, These sensors do not require intensive preprocessing for creating perspective / depth perception and can be thought of as a way to give the AI the ability to reach out into the real world. In this way we can bypass the need to replicate an artificial version of the visual cortex.

1.1 Theoretical framework

The idea of machines and supernatural beings having analogous abilities for interacting with the world as we do is not new. Legends and folk stories have carried this idea throughout time, and an idea can be a very powerful thing. History shows that if a conceptually strong idea arises in the human consciousness, it is only a matter of time until it will prevail and become reality. It is therefore intuitive to observe that plenty of research has been carried out towards face recognition if we take a look at scientific journals of psychology, computer vision, image analysis, pattern recognition and machine learning [Bruce and Young, 1986, Craw et al., 1987, Turk and Pentland, 1991b].

However, face recognition still remains an open problem [Zhao et al., 2003] and this is mainly due to the variance in the data — 2D images — such as expression, occlusions, illumination, viewpoint, aging, etc. Behavioral studies show that the recognition of identity and expression appear to proceed relatively independently [Humphreys et al., 1993]. The human brain has evolved to recognize a person's expression before the identity — this is intuitive since the first functionality increases the chances of survival.

1.1.1 Face recognition by humans

Even though the current research does not intend to replicate the human visual perceptual system for face recognition, it is of great value to understand how the human brain performs this complex task. Towards this goal, [Sinha et al., 2006] aggregates experimental results from behavioral and neuroimaging studies on humans, summarizing important clues towards machine face recognition.

Firstly, the article provides several clues that support the use of sensors that can directly capture the shape of the face instead of deriving it from 2D images: "human face recognition uses representations that are sensitive to contrast direction [. . .] pigmentation and shading play important roles in recognition." [Sinha et al., 2006, p. 1955, result 11]. Moreover, studies show that the human brain, even though sensitive to illumination direction, excels at generalization to novel illumination conditions.

This finding is in accordance with the current best performing face recognition methodologies (pseudo-3D) where computer vision algorithms [Blanz and Vetter, 2003] are used extensively to generate more data from one sample image by simulating variance in lighting conditions in order to infer a 3D model. This is an indication towards the idea that shape cues are very important and that a three dimensional representation of shape is necessary for achieving good results.

Regarding the two types of cues, pigmentation (color, texture, etc.) versus shape, behavioral studies reveal that they are equally important in the face recognition process [Sinha et al., 2006, p. 1953, result 9]. However, when shape cues in images are compromised (low resolution, noise, low light, etc.) the brain relies on color cues (pigmentation) to pinpoint identity. While this suggests that failure in human perception to infer shape would fallback to pigmentation cues, this lessens the requirement for redundancy — e.g. using color — in the case of depth sensors such as the Kinect, since shape is captured directly.

Motivation proceeding the use of video sequences in artificial face recognition is also encouraged by evidence in experimental studies. Intuitively, the human brain makes use of the temporal proximity of

images as "perceptual glue" [Sinha et al., 2006, p. 1955, result 13] for discernment of the whole face. It is quite likely that this is necessary for the brain to establish baseline shape cues such as the equivalent computational algorithms described in [Blanz and Vetter, 2003]. Additionally, it appears that dynamic cues such as the nonrigid facial musculature activity give clues about a structural piece of information: the adhesion points of ligaments. This information is more significant than simply having multiple viewpoints [Sinha et al., 2006, p. 1956, result 14]. Computationally, this implies proper segmentation and normalization (solving translation and rotation invariances) which isolates the nonrigid movement variance.

Another valuable result revolves around the strategy involved in face recognition (piecewise vs holistic): "Initially, infants and toddlers adopt a largely piecemeal, feature-based strategy for recognizing faces. Gradually, a more sophisticated holistic strategy involving configural information evolves." [Sinha et al., 2006, p. 1957, result 16]. This implies that features are learned first and then the focus shifts towards configural information and topology. Furthermore, "when taken alone, features are sometimes sufficient for facial recognition. In the context of a face, however, the geometric relationship between each feature and the rest of the face can override the diagnosticity of that feature. [. . .] configural processing is at least as important, and that facial recognition is dependent on “holistic” processes involving an interdependency between featural and configural information." [Sinha et al., 2006, p. 1951, result 4]. This suggests that the agreement between the configuration of the features of the face (alignment of eyes, nose, etc.) as well as the importance of each feature is just as important as the features themselves.

Representationally it appears that a "face space" emerges in the brain, which exaggerates shape deviations from a norm (dubbed as caricatures) in order to increase discriminability [Sinha et al., 2006, p. 1952, result 7]. This is also in accordance with another experimental result in prolonged exposure to faces which can cause "aftereffects", suggesting that the encoding strategy is prototype-based and the perception of faces is a highly plastic process which adjusts itself continually [Sinha et al., 2006, p. 1953, result 8]. Likewise, eigenfaces [Turk and Pentland, 1991b] and fisherfaces [Belhumeur et al., 1997] exploit a similar concept — yet most methodologies are focused mainly on the luminance information associated to the face image which is a naive approach towards storing structural properties. The illumination structure does not necessarily coincide with actual shape cues. When the same algorithm was applied separately for the complete color space [Torres et al., 1999], the machine recognition rate was increased by 3.39%. This further supports the idea that the illumination and texture information can not describe shape alone in a two-dimensioanl structure. The representation itself has to be able to connect observations from different viewpoints and as such has to be dynamic: invariant to pose and expressions. Hence, this was a clear hint towards generative models.

1.1.2 Fundamental methodologies primer

From an applied perspective, face recognition can be categorized in two subgroups, namely *face identification* and *face verification*. The former is a one-to-many dilemma and aims to recognize the identity of a person based on a set of already known identities in a gallery / dataset of face images. The latter is a one-on-one problem and attempts to evaluate whether a pair of faces is from the same subject or not. The research presented here is centered effectively on the former. If time is added as a constraint, then the most challenging face recognition application is biometric authentication [Jain et al., 2004] which implies live, simultaneous training and evaluation, sometimes using limited information. Although the existing systems are deployed mostly in controlled environments (such as video cameras in subways) they are limited to the environment in which they operate, hence they lack the ability to generalize. Data can be acquired from a live source, a movie clip [Wolf et al., 2011a] or any kind of digital photo [Huang et al., 2007]. The problem becomes more difficult if there is no control over the quality of the input data, which is usually not as consistent as in the datasets recorded in laboratory conditions [Sim et al., 2002, Phillips et al., 2000]. Hence recently, academic attention has shifted towards datasets which contain samples col-

lected from news articles on the Internet [Huang et al., 2007] or YouTube videos [Wolf et al., 2011a] which are considerably closer to realistic conditions due to the high variance in the data.

Consequently, most research in face recognition massively gravitates around the most available type of data — digital images, as it can be concluded from the thorough literature surveys [Zhao et al., 2003, Yang et al., 2002]. The authors conclude that the performance is limited by the variations in illumination [Chen et al., 2006] and pose. Algorithmically, there are two major schools of thought in traditional face recognition: appearance based methods and geometric methods. In a rigorous paper [Delac et al., 2005] evaluating 16 different appearance based algorithms, the authors conclude "Finally, it can be stated that, when tested in completely equal working conditions, no algorithm (projection–metric combination) can be considered the best". Other papers evaluating similar methodologies usually arrive at the same "no free lunch" conclusion [Draper et al., 2003].

As the research on purely appearance based algorithms such as the classical Eigenfaces or Fisherfaces [Turk and Pentland, 1991a, Belhumeur et al., 1997] has been exhausted and performance has reached an asymptote without equaling human efficacy, the attention has shifted towards geometric methods. It has become increasingly evident that shape cues are crucial to achieve high discriminability. A successful geometric method, which can be considered *pseudo 3D* employs three dimensional morphable models [Blanz and Vetter, 2003] : synthetic views are created for inferring the shape from texture and subsequently fitted onto a 3D morphable model in order to extract the deviations from the norm — which is an average face, constructed by acquiring 3D scans of faces.

The identification rates reported in [Blanz and Vetter, 2003] are promising: 95.0 percent for CMU-PIE [Sim et al., 2002] and 95.9 percent for FERET [Phillips et al., 2000] showing robustness across illumination and pose. However, the datasets are outdated and do not entail realistic conditions. Nevertheless, the concept of morphable models has been passed on to the purely three dimensional domain and will be discussed in the next section.

Recently, patch-based texture descriptors called Local Binary Patterns (LBP) [Ahonen et al., 2006], have been extensively used [Wolf et al., 2011b, Li et al., 2013] for feature extraction. In [Wolf et al., 2011b] piecemeal LBP histograms enhance the performance of face verification considerably while the accuracy rate is 58 percent on face identification using a subset of 50 people extracted from the Labeled Faces in the Wild (LFW) database [Huang et al., 2007].

The LFW dataset provides a thorough benchmark and realistic data — images of famous people extracted from news articles on the internet. It contains 13233 images of 5749 people, the number of examples per person varies but it is at least two. Thus, it has become one of the central hubs for face recognition research on digital images. However, even though there are several protocols for evaluation, all the methods report in fact *face verification* results (one-on-one pair matching). The results are constantly reported on their website [1].

A notable effort towards face recognition using simple human inspired V1-like features [Pinto et al., 2009] on the formerly mentioned dataset achieved a recognition rate of 79.35 percent, while the best performing algorithm [Li et al., 2013], using a fusion between LBPs and Scale Invariant Feature Transform (SIFT) [Lowe, 2004] and Gaussian mixture models [Reynolds et al., 2000] for matching, reported an accuracy of 84.08 percent. It is also worth noting that the best performing algorithm [Cui et al., 2013] on the less restricted protocol achieved slightly better performance — 89.35 percent — by learning face region descriptors using a sparse coding method similar to Bag of Features [Nowak et al., 2006, Lazebnik et al., 2006]. On the least restricted protocol — use of data from other datasets is allowed – the best performing algorithm [Chen et al., 2013] also uses high dimensional (100K) LBPs to achieve a verification rate of 95.17 percent. Support Vector Machines (SVM) [Cortes and Vapnik, 1995] are almost ubiquitous in all aforementioned research.

[1] Labeled Faces in the Wild Results: `http://vis-www.cs.umass.edu/lfw/results.html`

Effort towards pose variant non-holistic face recognition — which does not however take configural information into consideration — is the Bag of Words based research performed in [Li et al., 2010]. The face is split in 5×5 blocks and for each block SIFT [Lowe, 2004] features are computed and clustered in order to obtain a per-block dictionary. The block extracted histograms are concatenated to represent the face, thus increasing the discriminative power. The sparse representation ensures robustness towards facial expressions and rotation. Unfortunately, the authors do not report results on the newer LFW [Huang et al., 2007] or YT [Wolf et al., 2011a] datasets. Yet, the algorithm achieves the best recognition results on two other datasets: 98.92 percent and 100 percent on the AR and XM2VTS, respectively. The AR [Martinez, 1998] database consists of over 3200 face images of 135 subjects with variant facial expressions, lighting, and partial occlusions. The XM2VTS [Messer et al., 1999] database consists of 2360 face images from 295 subjects varying in poses, hairstyles, expressions and glasses. Both datasets are outdated and recorded in laboratory conditions.

Before the appearance of the LFW database in 2008, several datasets and benchmarks were driving the research community forwards. An independently administered technology evaluation of the capability of face recognition systems that try to meet requirements for large scale, real world applications — the face recognition vendor test (FRVT) [Phillips et al., 2003] reported that males are easier to recognize than females — the same being valid for older people compared to younger people. In the same article it was also reported that three dimensional morphable models [Blanz and Vetter, 2003] — an important landmark in face recognition research — and normalization increase performance, while video sequences [Gorodnichy, 2005, Zhou et al., 2003, Lee et al., 2003, Cohen et al., 2003, Sivic and Zisserman, 2003] only add a limited increase in performance while adding to the complexity of the problem.

While video-based recognition has mainly been performed on 2D holistic faces, sparse coding strategies have been extensively used for object recognition in video [Sivic and Zisserman, 2003, Nowak et al., 2006, Lazebnik et al., 2006] in order to deal with the fast search problem in abundance of data. These algorithms could be used in conjunction with high dimensional histograms in order to increase performance of face recognition from depth sensors such as the Kinect.

According to traditional face recognition methodologies [Zhao et al., 2003], non-holistic approaches generally have more flexibility and allow further classification possibilities based on different criteria such as sex, facial expression, gender, etc. This *modus operandi* suggests that the non-holistic processing of face parts (eyes, nose, mouth, eyebrows, etc.) — as observed in [Li et al., 2010] along with configural information should result in superior robustness and provide inherent redundancy and scalability. The findings from face recognition in humans [Sinha et al., 2006] also support these ideas, however, there has been yet no research that combines piecewise local feature processing with configural information.

A study investigating multimodal 2D and 3D face recognition [Chang et al., 2005] reports results on a dataset of 198 persons using Principal Component Analysis (PCA) [Jolliffe, 2005] methods which are used separately for each modality and combined for multimodal recognition. Their conclusions are that 2D and 3D have similar recognition performance when considered individually and combining 2D and 3D results using a simple weighting scheme outperforms either 2D or 3D alone. However, their methods are focused exclusively on PCA-based methods. The next section contains research on three dimensional data that goes beyond these methods and further improves performance.

1.1.3 Three dimensional techniques

Three dimensional face recognition has the potential to achieve better accuracy than its 2D counterpart by directly measuring the geometry of rigid features on the face. This avoids the pitfalls of traditional face recognition algorithms. The main difference between 3D face recognition and standard methodologies is that the type of information that is being processed contains shape information in addition to pigmen-

tation. Furthermore, the processing and computing of features is directly performed in $\mathbb{R}^3$ and is thus completely independent of the texture information.

Due to the formalization of the problem in computer vision research — a line of thought reminiscent of *divide et impera* — the general paradigm for face recognition research is focused around solving a pipeline of independent successive processes: normalization, feature extraction and classification. Since the transition from traditional to 3D data has been gradual, most of the established benchmarks for 3D face recognition contain datasets which contain complete frontal models (no occlusions) and thus eliminate the problems of normalization and missing information from the beginning. Hence, in this frame of thought, the bulk of research has been exclusively focused on feature extraction using computer vision methods. Thus, the remaining challenges in 3D face recognition revolve around normalization — variance in pose (rotation) and position (translation) as well as eliminating occlusions.

A thorough study [Abate et al., 2007] of the problems involved in 2D and 3D as well as multimodal approaches to face recognition after FVRT reviews the existing datasets and methods. The important conclusion is that each methodology has its own specific drawbacks and benefits and claims satisfactory recognition rates, however each algorithm is overtrained to a certain dataset and as such, a proper comparison is difficult. The study expressed the lack of a wide face database modeling real world conditions, in terms of differences in gender, ethnic group as well as expression, illumination and pose.

Due to the success of the FRVT [Phillips et al., 2003] and the motivation towards a common benchmark, a new dataset along with proper procedures emerged in the face recognition research. Published by the National Institute of Standards and Technology (NIST), it raised the bar in face recognition research and addressed almost all the problems of unconstrained face recognition, except pose variance and occlusions.

The Face Recognition Grand Challenge (FRGC v2.0) [Phillips et al., 2005] was formulated as a series of six challenging experiments. The dataset was periodically updated, becoming increasingly more difficult, finally containing both 3D scans and high resolution still images taken under controlled and uncontrolled conditions. There are 4007 scans of 466 people containing both range data and texture, acquired with a Minolta Vivid 900/910 series sensor which has a resolution of 640×480 pixels. The scans contain various facial expressions, subjects are 57 percent male and 43 percent female with a majority of age distribution between 18 and 22 (65 %). The 3D scans exhibit various facial expressions, are normalized (frontal pose) and do not have accessories such as glasses nor contain occlusions.

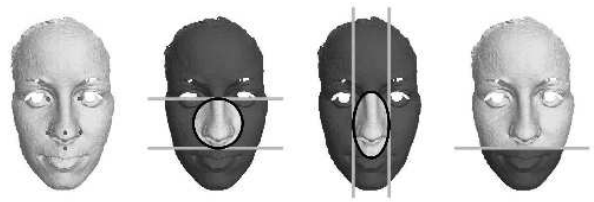


Figure 1.1: Similarity measures are computed independently for explicitly defined face regions and combined using ensemble methods. From [Queirolo et al., 2010]

The algorithm with the best reported results [Queirolo et al., 2010] on the 3D benchmarks within FRGC performed registration (fine tuned alignment) of the frames by Simulated Annealing (SA) [Kirkpatrick et al., 1983]. Using a surface interpenetration measure — a Mean Squared Error (MSE) of the interleaving of two depth images — a verification rate of 96.5 percent with a false acceptance rate (FAR) of 0.1% and a rank-one accuracy of 98.4 percent was achieved in the identification scenario. Informally, faces were overlapped as much as possible and the shape differences used for classification.

Initially, registration was performed considering the shape of the whole face. Since the dataset contains expression variant scans, the alignment was subsequently improved by taking into consideration only the forehead, eyes and nose regions. Furthermore, the best results were achieved by combining similarity scores from several face regions as illustrated in

fig. 1.1 — left to right: the entire face region, the circular area around the nose, the elliptical area around the nose, the upper head (eyes, nose, and forehead). It should be noted that the algorithm used for the keypoint identification and segmentation of the aforementioned face regions is not robust to variance in pose.

While the overall results are the best, when experiments are performed independently on the non-neutral face expression set, they report that methods using deformable face models perform better and achieve a higher robustness towards facial expression. Not surprisingly, in [Kakadiaris et al., 2007] the authors report the second highest recognition rate of 97.0 percent rank-one on the 3D FRGC v2.0 dataset. Deformable face models [Metaxas and Kakadiaris, 2002] are used for fitting a 3D model to the face scans in order to achieve expression invariance. The differences are converted into a geometry image (2D projection) and a normalized 3D model (X, Y, Z) and subsequently decomposed for comparison using Haar wavelet transforms. Their algorithm requires 15 seconds for initial registration and model compression and can process 2000 observations per second on a regular computer at the time of writing. The idea is very similar and probably inspired from the three dimensional morphable models in the hybrid 2D + 3D method [Blanz and Vetter, 2003].

From the same paper, results on an in-house recorded dataset UHDB11 [2] which contains observations with pose variance, occlusions and accessories is also reported and as expected has a lower recognition rate of 93.8 percent. In this experiment, pose normalization is performed using spin-images for solving the initial coarse translation variance followed by ICP [Besl and McKay, 1992] for solving the rotation variance and finally fine-tuned using SA [Kirkpatrick et al., 1983]. Yet, the mean deviation from frontal pose in their recorded dataset is not specified.

Several other methodologies from the results on the FRGC [Phillips et al., 2005] 3D benchmarks present valuable research towards non-holistic face recognition. A fusion [Faltemier et al., 2008a] of 28 spherical face regions are combined by consensus voting and achieved a verification rate of 93.2 percent at 0.1% FAR and rank-one identification of 97.2 percent. A modified version of the ICP [Besl and McKay, 1992] algorithm was used for registration and provided the MSE as a discriminative measure. It is most likely that the recognition rates were outranked since ICP is known for its slow convergence and tendency to drift towards a local optima. Furthermore, while ICP might be a good strategy for coarse registration of two scans, it certainly does not have the best discriminative capacity. Despite its inherent disadvantage, this ensemble method achieved a high ranking result.

Another non-holistic approach [Lin et al., 2007] reveals that using Linear Discriminant Analysis (LDA) for defining the weights for the sum rule of ten combined face regions improves face recognition rates considerably. It has been demonstrated [Kittler et al., 1998] that the sum rule outperforms other classifier combination schemes. The novelty of this research resides in the proof that further optimization of the sum rule can be achieved for increasing recognition rates. This method achieved a 90.0 percent verification rate at 0.1% FAR.

A multimodal approach is presented in [Mian et al., 2007] which combines 2D SIFT local features with 3D similarity measures computed independently for two face regions — forehead + eyes and nose and a holistic global face descriptor. From the experiments it can be observed that the holistic encoding performs the worst, followed by a slight performance increase in SIFT while the best performance is achieved using the combined forehead and nose regions. Thus, it is clear that in the absence of deformable face models [Metaxas and Kakadiaris, 2002] the rigid parts of the face are key to achieving expression invariance, however this limits the recognition rate since not all face shape information is used. SIFT has the capacity for detecting salient keypoints, yet it does not have the discriminative power to capture the inter-face space in its default implementation. Perhaps a better method would have been the pyramid matching scheme used in [Lazebnik et al., 2006]. Considering only results for the 3D face images, a

[2] UHDB11 dataset: `http://cbl.uh.edu/URxD/datasets/`

verification rate of 86.6 percent was attained at 0.1% FAR. When using a neutral gallery and all remaining images as probes, this verification rate was increased to 98.5 percent at 0.1% FAR.

An expression-invariant multimodal face recognition algorithm was proposed by [Bronstein et al., 2005] which considers the 3D face surface to be isometric. The effects of expressions are eliminated by finding an isometric-invariant representation of the face. Unfortunately, this process also "softens" the details of some key features such as the shape of the nose and eye sockets. Furthermore, a database of only 30 subjects was used and extreme facial expressions (such as an open mouth) were not discussed. However, the authors demonstrated that surface reconstruction (using algorithms such as Poisson reconstruction [Kazhdan et al., 2006] or Marching cubes [Lorensen and Cline, 1987]) is not a good research path for face recognition and should be completely avoided. The same authors [Bronstein et al., 2003] further confirmed the argument of [Zhao et al., 2003] and the findings in human face recognition [Sinha et al., 2006] that combinations of texture from 2D and 3D shape information "could potentially offer the best of the two types of methods" [Zhao et al., 2003].

The Rotated Profile Signatures (RPS) [Faltemier et al., 2008b] nose detection algorithm acknowledges the fact that 3D scans can contain missing information at extreme pose angles and is capable of performing nose detection with an accuracy greater than 96.5 percent. The authors thus address the need for a dataset which contains actual pose variance (in contrast with artificially rotated frontal scans) and provide a method of finding the nose tip. The experiments were performed on the NDOff2007 [3] of 3D faces acquired under varying pose which contains over 7300 total images of 406 unique subjects. The changes in yaw range from -90° to 90° and changes in pitch range from -45° to 45°. This research thus addresses the problem of face recognition in the presence of non frontal head pose, which is still a challenge that current best performing algorithms on FRGC presented above have not yet considered due to the nature of the dataset in the benchmarks.

Since the research presented here uses the Kinect sensor it is sensible to mention the current system used for player identification in the Xbox 360 console. The Kinect Identity [Leyvand et al., 2011] software is able to recognize two "players" simultaneously, however, they combine multimodal information, such as height and shirt color, yet they do not specify whether they use 3D data for face recognition. In the recently announced Xbox One console they advertise that they can distinguish between six "players" simultaneously. Lately, findings from cognitive neuroscience have made multimodal biometric systems become more popular, such as face recognition combined with voice, fingerprint or iris recognition [Jain et al., 2005, Snelick et al., 2005] however, these findings are not discussed here as they are outside the scope.

1.2 Research questions

As previous research has demonstrated, the problem of face recognition has been addressed from various perspectives and it has often been reduced to simpler forms compared to what real world conditions would imply. This is most likely due to the fact that much of the 3D face recognition research has been heavily influenced by the nature of the datasets that have been published, driving the scientific research forwards on a very similar path. As such, the modeling of real world conditions has been incremental (for example FRGC has been iteratively updated, it did not contain 3D scans initially) and thus the algorithms have been overfitting over the same traditional computer vision paradigm which implies a computationally intensive precise normalizing step, which limits the real-time operation on standard hardware platforms.

[3] NDOff2007 dataset: `http://www3.nd.edu/~cvrl/CVRL/Data_Sets.html`

Only recently, the NDOff2007 [**?**] dataset, which contains actual pose variations — and thus authentic partial observations / occlusions — has been published. This database thus models real world conditions accurately, however the quality of the recorded data is much higher than the data coming from a cheap, off-the-shelf depth sensor such as the Kinect 360 [4] or the Asus Xtion Pro [5]. The emergence of these new types of affordable and compact infrared depth sensors has given birth to new opportunities for research. This type of sensors, whether economical and compact, also come with a drawback — noisy data with decreasing resolution as distance to the sensor increases. To our knowledge, there has been no published research on the topic of real-time 3D face recognition in unconstrained environments, using such commodity depth sensors.

Therefore, the first question we have to ask is if face recognition using cheap off-the-shelf sensors is even possible, and if so, what are the limitations of using such sensors? Towards answering these questions, it is necessary to analyze the conditions which can influence performance. As such, a dataset consisting of isolated types of variance sets (pose, distance to sensor, expressions) was recorded.

An important conclusion that can be drawn from the aforementioned literature is that virtually all best performing algorithms are non-holistic. This is also consistent with the behavioral and neuroimaging studies on human face recognition presented in [Sinha et al., 2006]. We have observed that the performance is highly dependent on the precise pose normalization, in order to generate clear-cut overlapping face regions. Moreover, in all cases the face regions are explicitly defined and the authors do not provide an experimental motivation for the region selection method.

Hence, the next question is whether the identification of face regions can be done unsupervised. While for pose normalized data this is trivial, we do not intend to focus on the problem of normalization and therefore propose to investigate whether it possible to sample consistently for pose variant data using a viewpoint invariant sampling method such as SIFT [Lowe, 2004].

While the article that started it all [Turk and Pentland, 1991b] was based on human studies, the latest face recognition research has been focusing towards solving what were the main engineering drawbacks of that method (pose normalization), instead of centering research on the key discovery of using dimensionality reduction, hence sparse representations. This leads us to conclude that non-holistic generative models are paramount to achieving realistic and computationally tractable solutions.

Considering the research on human face recognition and the current scientific developments we therefore focus on incorporating as much variance as possible within the classification model. This brings us to our next questions: what is the right scale and resolution for a piecewise representation of data, given the occlusions and the dynamic nature of faces? What are the limitations of discriminative models? What is the ideal abstractization level for such a task?

1.3 Scientific relevance

All our machines are merely tools — extensions of ourselves. Just as binoculars are an extension of human sight, books are an extension of human memory and communication and just as a pair of pliers is an extension of the human hand, the computer is an extension of the human mind. We are moving closer all the time — getting into the machine.

As technology is more and more part of our lives, we ought to make sure that machines adapt to us, not *vice-versa*. This research would make a step towards a better understanding of ourselves and the way we interact with machines, loosening this gap. Communication and interaction with our tools would be

[4] `http://www.xbox.com/en-US/xbox360/accessories/kinect/KinectForXbox360`
[5] `http://www.asus.com/Multimedia/Xtion_PRO/`

more natural in general and also tailored to each individual's personality. It would only be natural to model technology towards our preference instead of adapting ourselves to the technology.

We are witnessing an increasing trend: the appliances and gadgets we interact with daily are becoming more and more ubiquitous. Still, they lack satisfactory personalization since in practice this requires an adequate means of perception for the machine. Seamless detection of a person's identity or psychological state would result in novel methods of interacting with computers where the user would interconnect in a natural manner. This, of course, requires that the algorithms are robust in unconstrained environments.

The field of robotics is expanding, moving from factories into society and ultimately our homes. It is perhaps most evident here that robust face and expression recognition would facilitate a better communication channel between robots and humans.

There is a great deal of research devoted to the subject of face recognition and biometrics in general, however most methodologies for identification using only the face have been only applied to offline face recognition tasks. There is yet no algorithm that performs this task accurately in unconstrained environments. Since there have not been any substantial discoveries recently, research towards face recognition using commodity depth sensors will hopefully stimulate the scientific community.

Chapter 2

Data acquisition and representation

Given our goal of using the depth sensor in unconstrained environments it is crucial to identify its physical limitations and employ robust machine learning strategies which can cope with the inherent circumstances. Consequently, a good strategy to avoid an exhaustive, slow exploration of the experimental search space is to start off with a very isolated, easy to solve problem, then gradually increase the difficulty. Hence, the recorded data reflects the same idea: data sets contain separate variations in angle, translation, depth, etc or combinations thereof.

This in turn simplifies the problems of normalization and missing data, which will also be discussed in more depth in the following sections. Then, the first step is to keep only the data that is relevant to our objective, thus it is mandatory to isolate the face from the rest of the 3D scene. Our intention is to capture as much variation as possible in face expression while keeping a relatively low diversity in translation and rotation.

2.1 Anatomy of an infrared depth sensor

The popularity of infrared depth sensors such as the Kinect is increasing along with the video game market. It is therefore sensible to assume that the technology will further improve, resulting in higher resolution sensors with an even more compact form. At the time of writing, the new Kinect for Xbox One was released and is reported to have a resolution which is almost 7 times higher than the model used in this research.

Originally the sensor was developed for natural interaction for players in various games, however, due to the low cost and compactness it became an interesting alternative to expensive laser scanners for applications such as indoor mapping, gesture recognition or 3D modeling. An important feature of this type of sensors is that depth data can be recorded regardless of the lighting conditions, which entails that shape cues can be extracted even in complete darkness. The major drawbacks with this type of sensors are that the resolution decreases along with distance to the sensor and that the depth information contains "noise" due to measurement errors [Khoshelham and Elberink, 2012].

2.1.1 Technical aspects and limitations of the Kinect sensor

According to the manufacturer's specifications [1] the field of view for the Microsoft Kinect for the Xbox 360 gaming platform is 57 degrees on the horizontal plane and 43 degrees on the vertical plane. Both color and depth can be recorded at a maximum rate of 30 frames per second (FPS). The Kinect projects invisible divergent infrared light (fig. 2.1, right) in the environment while a monochrome CMOS (complimentary metal-oxide semiconductor) sensor measures its "time of flight" (TOF) after it reflects off the environment. The time of flight principle is the same as with sonars: by knowing how long it takes for the projected infrared light beam to return to the sensor, given the speed of light, it is possible to measure how far an object is.

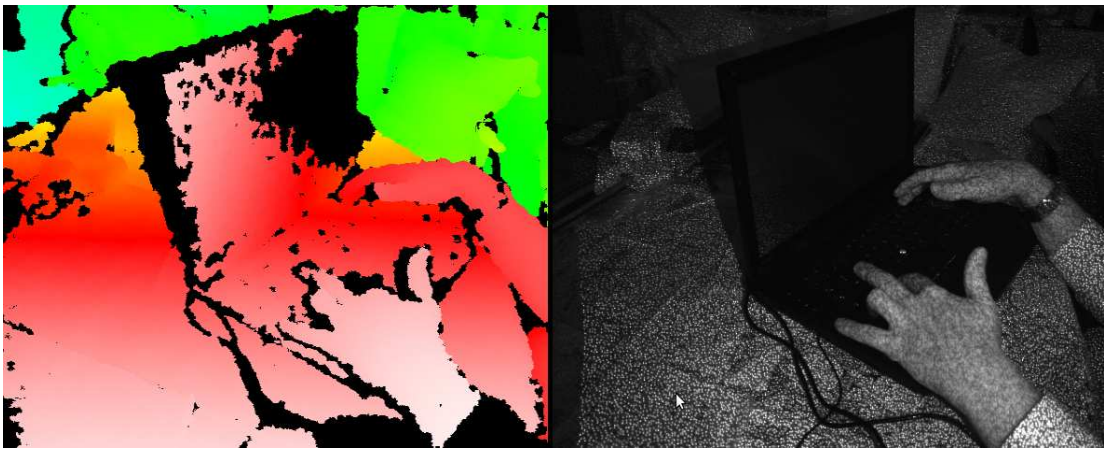


Figure 2.1: Left: a visualization of depth information, white objects are close, red is medium range and green is far. Right: the grid of infrared light points projected into the environment.

The sensor emits a "grid" of multiple beams and it can measure the distance to $640 \times 480 = 307200$ locations. Even though the number of points in each frame is constant, the distance between the points varies due to the divergent nature of the infrared light emitted. Hence, the density eq. (2.1) is inversely proportional to the square distance from the sensor. Because of this relationship between the distance Z to the sensor and density ρ, a disparity measure d — which appears due to the displacement between the emitter and receiver — should be computed for each frame. Since the actual disparity measures are not streamed due to lack of bandwidth they are normalized as d'. These inaccurate measurements and the rounding of values cause depth measurement errors (fig. 2.1, left). The variance of depth measurements eq. (2.2) is thus proportional to the square distance from the sensor to the object.

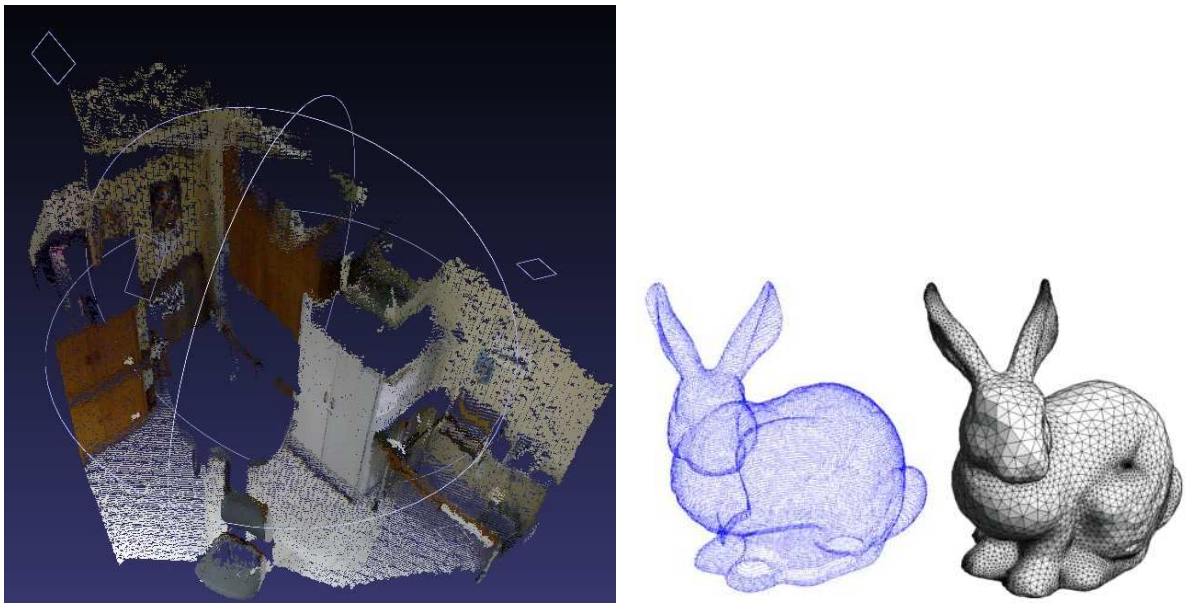
$$\rho \propto \frac{1}{Z^2} \qquad (2.1) \qquad\qquad \sigma_Z^2 = \left(\frac{\partial Z}{\partial d}\right)^2 \sigma_{d'}^2 \qquad (2.2)$$

The experimental measurements reveal that the random error of depth measurements increases quadratically with the distance from the sensor, explicitly [Leyvand et al., 2011, fig. 10, p. 138] it is only a few millimeters at 0.5 meters from the sensor ~0.25 cm at 1 meter from the sensor and almost reaches ~0.5 cm at 1.5 meter from the sensor. The highest recorded error is 4 cm at the maximum range of 5 meters. Therefore, due to the decreasing density of point clouds and the noise in the measurements, for the current objective, the recorded dataset will be constrained to a maximum distance of 1.5 meters from the sensor.

[1] Microsoft Kinect 360 Manual: `http://support.xbox.com/en-US/xbox-360/manuals-specs/`

2.1.2 The sensor data: point clouds

Since each pixel contains additional depth information, this type of sensors are sometimes called RGB-D (Red, Green, Blue, Depth). The common name in the literature of a collection of RGB-D pixels – *point cloud* — comes from the visualization technique: color pixels are spread out in a cloud-like 3D scene as can be observed from fig. 2.2(a).



(a) Vizualisation of a color point cloud in the MeshLab open source software (`http://meshlab.sourceforge.net/`)

(b) Stanford bunny — point cloud on the left, triangle mesh on the right

Figure 2.2: Examples of data representation

Point clouds are usually defined by X, Y, and Z coordinates and most commonly represent the external surface of an object — which can be textured or not. The Kinect sensor concatenates the depth measurements for each "distance to sensor" pixel with the color information obtained at the same position in space in order to obtain points which contain a 3D coordinate along with color components. Each pixel has a position relative to the origin $O(0, 0, 0)$ — in our case, the sensor — and also the hue information, typically represented in the RGB color space, as it can be observed in fig. 2.2(a).

The point density can be increased by registering (overlapping) consecutive depth images using algorithms such as Iterative Closest Points (ICP) [Besl and McKay, 1992]. While this is quite useful for mapping purposes, where the objects are rigid and usually do not change location, when dealing with non-rigid surfaces such as faces the procedure becomes considerably more difficult. This is due to the fact that not only the location of each pixel changes for each frame, but the pose, angle towards the sensor origin or the expression can change as well — which usually results in a lower consensus measure between two frames.

This abstractization level is therefore used to store the data. Even though the resolution might change according to the distance to the sensor, the overall shape and size of objects regardless of the relative position to the sensor, is kept the same. Furthermore, the representation is simple, can be stored as ASCII text and therefore facilitates easy importing and exporting to any software framework or between programming languages.

2.1.3 Partial observations and internal models

Even though point clouds can be directly visualized, they are most commonly converted and represented as polygon or triangle mesh models (fig. 2.2(b)) using surface reconstruction algorithms. Poisson surface reconstruction [Kazhdan et al., 2006] is such an algorithm which can robustly recover fine detail from noisy data and thus produces highly detailed surfaces. However, in the case of non-rigid objects such as faces, such an algorithm that can integrate information over time, will only produce distorted faces due to expressions. Furthermore, this kind of algorithm requires intense processing, thus imply a long registration procedure, while when recognizing persons, the data coming from the sensor would still be noisy in comparison with the internal model.

According to the pose of faces relative to the sensor origin, the information captured from the sensor can contain only partial observations. When a person is looking towards the right and is exactly in front of the camera, then the infrared beams will only fall on the left side of the face. This will result in a point cloud which will contain the shape of only half of the face as illustrated in fig. 2.3. The various configurations of position and pose and/or occlusions will thus result in frames with missing data. Therefore, due to the type of observations and the animated nature of faces, the current research will not further investigate surface reconstruction algorithms and will consider each frame as an independent observation, saving each frame in point cloud format, each of them containing pixels with coordinates and color values.

2.2 Recording a series of datasets

"The only source of knowledge is experience." — Albert Einstein.
When recording a dataset, it is crucial to capture observations which, when taken as a whole, ideally contain all the possible variations which can occur and thus describe the subject in all deviations from a norm. In turn, the performance of a machine learning algorithm depends on the "richness" of experience. First of all, in the case of 3D face recognition it is necessary to consider the degrees of freedom that are involved. The first is the position of the head relative to the camera. This is restricted by the field of view of the sensor as described in the previous section. The other is the rotation of the head around all three axes: roll, pitch and yaw — an aspect that has not been considered in any widely used dataset. Therefore, the different combinations of pose and position to the camera generate distinct observations. Furthermore, since the face is not a rigid object another type of dissimilarity arises. While this makes the problem considerably more difficult, it is the only way in which natural interaction with a computer can be achieved.

Given the physical capabilities of the sensor, the experimental studies on humans regarding which types of cues are most important, the variations which should be eliminated and the representational scheme, a series of datasets were recorded in such a way as to contain only one type of significant variance per each set: rotation (roll, pitch, yaw), distance to the sensor, expression and finally an unconstrained set.

Thus, for each person, six separate datasets were recorded — the final one is indeed the most realistic since no constraints are imposed on the position of the subject nor on the rigidity of the face. Further combinations were possible, such as repeating the rotation procedure in different positions relative to the sensor, however that would have required considerable more time for recording the datasets. Regardless, this type of variation is captured within the stochasticity of the positions in the final unconstrained set.

2.2.1 Procedure

The main goal of the dataset recording procedure was to capture and isolate changes in the extrinsic variance such as head pose, expression and illumination while capturing the intrinsic parameters such as the shape of the face and the texture information. For the recording of all sets, the subjects were seated in front of the sensor and the height of the chair was adjusted in order to have the nose initially pointing at the sensor. The distance to the sensor was kept at approximately 0.6 meters, except for the z translation and the unconstrained set.

For the first four datasets — I yaw, II pitch, III roll, IV z translation — the subjects were asked to keep a neutral facial expression and not talk during the recording procedure in order to focus on the problem of missing data due to camera viewpoint and normalization (translation and rotation). As it is intuitive to determine, for set I the subjects were asked to move their heads from left to right and *vice-versa*.

For set II the subjects were asked to move their heads up and down, while for set III the subjects were asked to tilt their heads either left-right as much as possible. In all three cases the maximum angle was $\pm 45^{\circ}$ in either direction. For set IV the subjects were asked to keep the X and Y position of the head fixed while the chair was moved towards and away from the sensor. The range in movement along the Z axis was between 0.4 meters and 1.3 meters.

When recording set V the subjects were asked to talk and display various facial expressions while keeping the position of the head relative to the sensor constant for all subjects. The same procedure was repeated for the unconstrained set VI, only that the sensor was moved in a circular manner around the head, starting far from the subject and gradually getting closer. The angle of the camera towards the face was also changed during the circular movement. In all cases the movements were slow and incremental in order to capture the gradual changes in pose, translation and expression. This resulted in a varying number of observations per person per set.

2.2.2 Head detection and tracking

While the current research does not focus on pose normalization, some preprocessing steps were necessary in order to capture only the face shape from the entire point cloud, while in the process also eliminate noise and unwanted artifacts. These steps were performed while recording the datasets with an average frame-rate of $1.5Hz$, which can be further optimized, however this was not the main objective.

A real time algorithm for head detection and pose estimation using regression forests [Fanelli et al., 2011] implemented in the Point Cloud Library (PCL) [2] provided the position of the head in the form of a centroid and a vector originating from the centroid describing the estimated head pose. The algorithm is based on discriminative ensemble random regression trees which first discriminate which parts of the sensor data belong to a head, subsequently using only those patches to cast votes for the final position and pose estimate.

During the training procedure the goal was the minimization of the entropy of the distribution, head position and orientation of the patch class labels (the dataset was annotated). However, each frame is processed independently, which sometimes results in oscillations of the position of the head or the pose, even if the pose is not changed.

[2] The Point Cloud Library: `http://www.pointclouds.org`

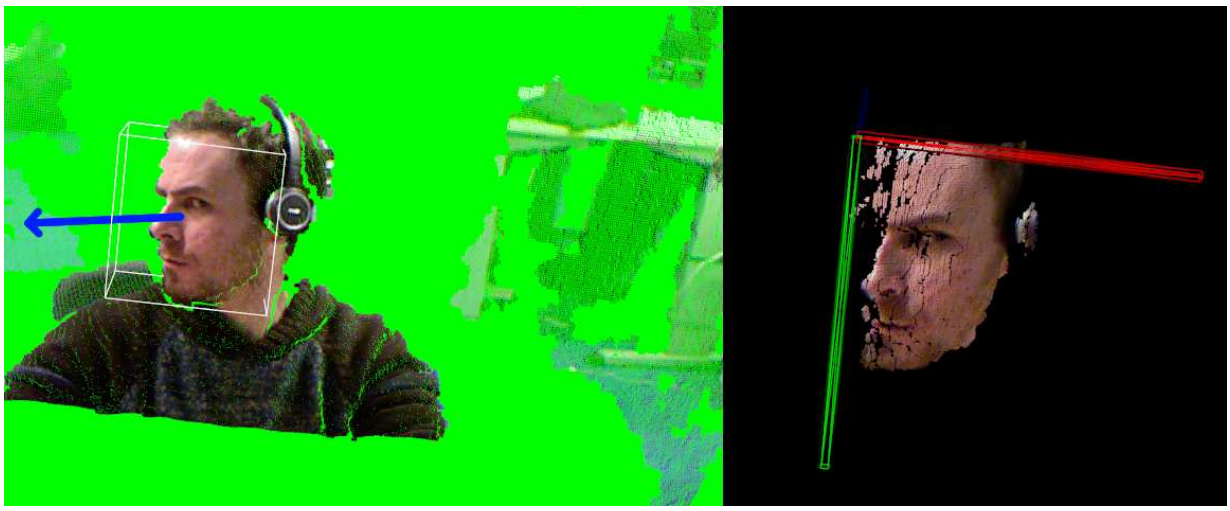


Figure 2.3: Left: zoomed in original point cloud. The cube delimits a fixed size volume around the head. The blue arrow is an estimation of the pose. Right: the data inside the cube is cropped, then rotated towards a frontal position and finally translated towards the origin.

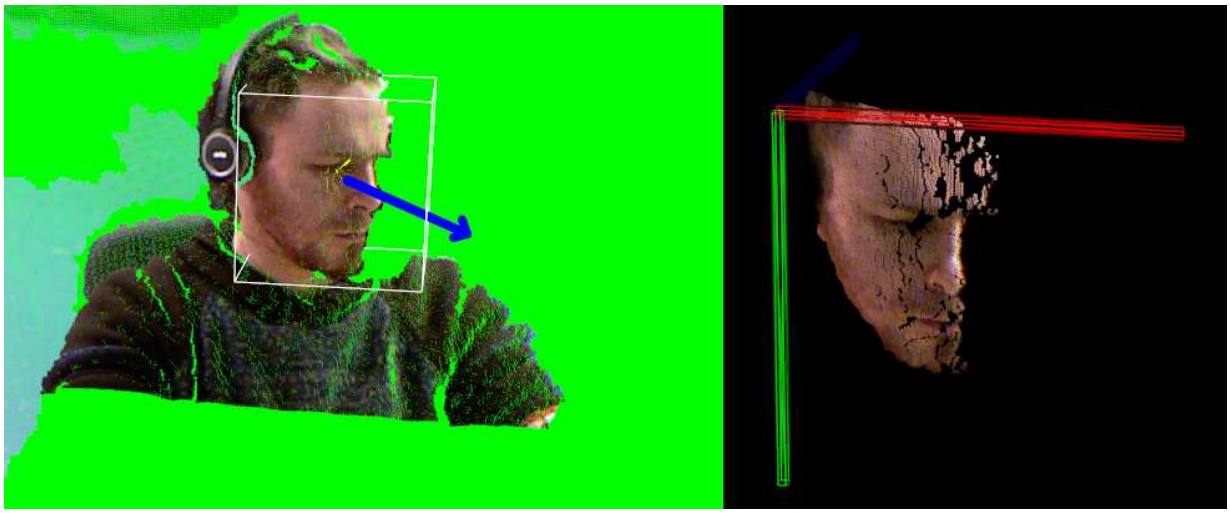


Figure 2.4: The same procedure is applied as in the previous figure. Since the pose is different, the cropped face is more closely aligned to a common canonical form, since there are points close to the YZ plane. The two cropped faces from figure 2.3 (right) and the current ones do not overlap.

2.2.3 Segmentation, rotation and translation

A fixed size cube with the edge of 0.1 meters with its centroid set to the position of the head returned from the head detector was used to crop the face. The cube's yaw, pitch and roll was kept fixed, with the front face parallel to the x,y axes at all times, thus it could only perform translation movements. In fig. 2.3 the cloud viewpoint was changed, the cube seems slightly tilted, however it can be observed that the white edges of the cube are parallel to the axis on image on the right.

After the segmentation of the head, the cropped data is rotated in space according to the pose estimation vector such as to achieve a frontal face view towards the camera. The blue line is an estimation of the pose, however it should be noted that this vector oscillates even when the pose is not changed. At this point, the coordinates of each point are still in the context of the whole scene, in other words, the point cloud's position is still relative to the sensor — the origin. Since we are interested only in the distribution and position of the points in the cropped region, the point-cloud is translated in space as close as possible to the origin $O(0,0,0)$. This is done by first searching for the minimum distance to each axis $Min(P_{axis})$, then independently subtracting these three minimum values per axis from the corresponding coordinate component of each point, see eq. 2.3.

$$\vec{P}_{axis} = \vec{P}_{axis} - Min(P_{axis}) \quad axis \in [X, Y, Z] \tag{2.3}$$

The result can be seen in fig. 2.3 on the right side of the image. The pose is approximately frontal, however due to the fact that there is no data on the left the whole cloud is shifted too close to the axis. If we were to overlap a frame with a different pose such as the one in fig. 2.3 with the one in fig. 2.4 then there would certainly be an inconsistency in translation. The solution to this problem is described in the following paragraphs.

2.2.4 Preprocessing

The segmented face sometimes contains disconnected clusters of points such as hair, parts of the neck, earrings, etc. In order to remove these clusters, euclidean clustering was used. The algorithm creates a Kd-tree representation of the point cloud and iterates over each point, checking the distance to its neighbors in a radius smaller than a threshold value. It then assigns the point to that cluster and the iteration continues with the unvisited points. The threshold value was set at 2.2 times the resolution of the point cloud. It is necessary to compute the resolution of each frame, since the projected beams of infrared light are divergent and there are less samples further away from the sensor. As such, the resolution of the cloud changes according to distance to the sensor. The resolution is determined as the average distance between the points in the cloud. After the segmentation process is finished the clusters which do not contain at least 1000 points are removed. While this does not eliminate all artifacts it results in frames with less irrelevant data.

2.2.5 Coarse normalization

While great progress has been observed on face alignment, building a robust normalization system is a very challenging task by itself and requires a lot of engineering efforts. As a result, state-of-the-art pose normalization systems, are often not fully accessible to the research community. A quick overview of the methods used in the best performing algorithms in the Face Recognition Grand Challenge (FRGC) reveals that most of the approaches follow the standard method which involves normalization, feature extraction, classification. Hence, ICP [Besl and McKay, 1992] is used for pose normalization, some also use it as a discriminative measure [Faltemier et al., 2008a, Mian et al., 2007] while sometimes Simulated Annealing (SA) [Kirkpatrick et al., 1983] is used for further fine tuning the pose, with much better results, as described in [Queirolo et al., 2010, Kakadiaris et al., 2007].

While the current research does not consider normalization as a crucial or necessary step, a coarse normalization step was performed in order to capture data and analyze recognition performance in the presence of reduced pose variance for each type of rotation, as described previously in the recording procedure section. The PCL implementation of the head detector also provides an initial pose estimate. However, it

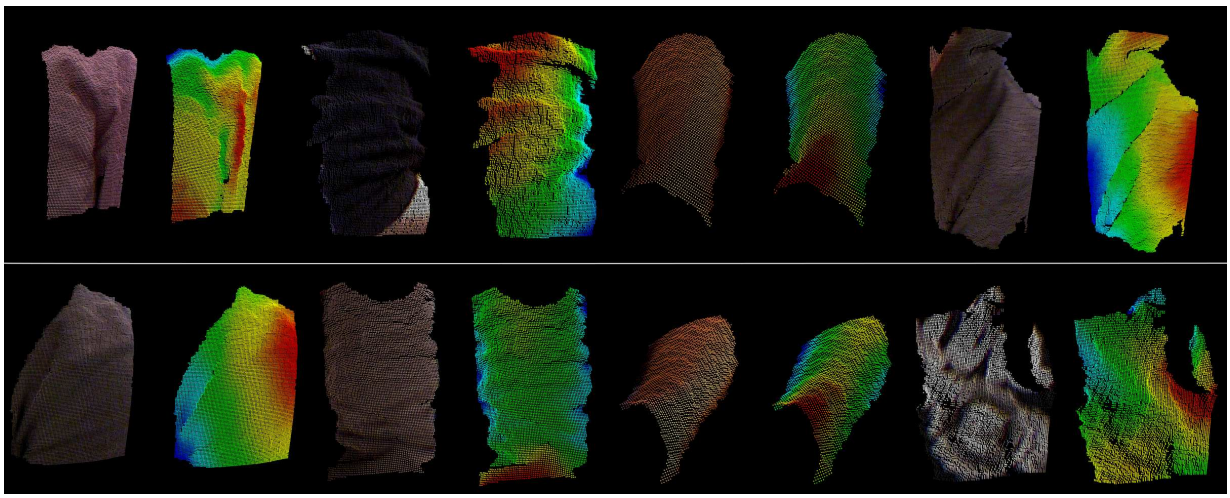


Figure 2.5: False positives. The images are depicted in both original texture and heat-map (red is close, blue is far). Some of the items are shoulders, elbow regions, arms, folded clothing, etc.

stochastically returns false positives (non-faces) such as those depicted in fig. 2.5. Furthermore the pose estimate is not always accurate, hence the detection process is followed by few steps of ICP in order to further minimize the rotation variance and reduce the number of false positives. As observable from fig. 2.5, it is most likely that the false positives are due to the similar curvature of the surface. Perhaps an additional filtering by texture would improve detection.

Figure 2.6: Generic 3D face template used for alignment to a common canonical position during ICP.

ICP iteratively revises the transformation (translation, rotation) needed to minimize the distance between two point clouds. A downsampled generic 3D face model (fig. 2.6) was used as a target norm for the alignment. The average 3D face was placed as close as possible to the origin as described earlier (eq. 2.3).

Initially, the distance is computed using a spatial nearest neighbor search for finding correspondences, after which the transformation parameters are estimated by using the Mean Squared Error (MSE) cost function.

If the initial alignment error returned by the cost function is above a qualitatively determined threshold of 0.00035, then further processing of the current frame is immediately stopped and the frame is discarded. This further reduces the number of non-faces and speeds up processing.

Otherwise, if the initial alignment score is valid, the transformation is applied and the point association step is repeated until either of the stopping conditions are met. These were set to a maximum of 50 iterations and a target value of of 0.0001 for the MSE. In order to increase time performance and achieve satisfactory results, ICP was applied to a uniformly sub-sampled collection of points from both the frame and the target.

2.2.6 Dataset properties

Following the previously detailed procedure and using the pipeline described above, the dataset was recorded and contains a total number of 18 subjects, out of which two were recorded twice, also wearing glasses. Since the PCL framework was used for the head detection implementation and preprocessing, the file format for each observation is also ".pcl" since it provides a clean, intuitive way to transfer the data to other environments and also provides an easy way of editing the data manually.

For each observation / frame the data is stored in form of an ASCII text file containing a matrix with columns X, Y, Z, RGB — each line thus having the coordinates and color values for one point. The color information was kept although it is not used here. Each observation / text file has a variable number of points. In total there are 4675 observations captured over all sets, with an average number of frames per person of 260. The total number of frames per class and per subset along with the mean and standard deviation is given in table 2.1.

The data was structured hierarchically: for each subject a new folder was created and renamed using the first name and first letter of the surname separated by an underscore. The folder name is thus the label of the class. For each subject six sub-folders were created labeled as "0_yaw", "1_pitch", "2_roll", "3_z_translation", "4_expressions", "5_freestyle". In each one of these sub-folders there are a variable number of observations stored as ASCII ".pcd" files. The remaining false positives were manually moved to a separate folder.

Even though the dataset was recorded with color there were no procedures during the recording in order to vary the lighting conditions. As such, the illumination conditions are the same for all subjects, except for the 5th dataset in which the camera was moved around the subject while the subject was free to move as well. While the color information could be used, it is certainly far from real-world conditions since there was not enough variance to be captured.

Class label	Yaw	Pitch	Roll	Z-Tr	Expr	Free	# Obs	μ	σ
albert_e	19	24	18	51	27	112	251	42	36
amir_s	15	43	17	50	55	54	234	39	18
amir_s_glasses	30	19	28	26	81	23	207	35	23
anton_m	29	52	17	30	27	35	190	32	12
ayla_k	18	15	13	29	27	70	<u>172</u>	29	21
ben_w	61	59	30	52	45	68	315	53	14
davey_s	26	34	20	20	33	46	179	30	10
diederik_v	18	22	9	37	23	99	208	35	33
florin_s	39	42	31	46	52	131	341	57	37
joost_b	32	22	13	51	30	87	235	39	27
marcel_b	63	26	33	31	35	62	250	42	16
marco_w	15	19	19	50	33	111	247	41	37
marko_d	63	46	65	63	43	75	355	59	12
marko_d_glasses	47	42	15	78	71	197	<u>450</u>	75	64
niels_v	33	39	33	48	40	23	216	36	8
rick_m	24	13	22	29	23	128	239	40	44
roald_b	46	62	49	58	37	56	308	51	9
sybren_j	32	29	20	105	15	77	278	46	36
Total / subset	610	608	*452*	854	697	*1454*	**4675**	779	356
μ	32	32	24	45	37	77	260		
σ	17	17	15	23	19	46	71		

Table 2.1: Number of observations per class and per subset. The minimum and maximum are underlined for classes and slanted for the subsets. The total number of camera observations is in bold text.

Chapter 3

Ensemble of unsupervised face region experts

Appearance based holistic face recognition algorithms [Turk and Pentland, 1991a, Belhumeur et al., 1997] are obsolete and can not describe nor separate the variance accordingly. Analyzing the evolution of the methodologies with the best performances participating in the FRGC [Phillips et al., 2005] one can observe an increasing sophistication in the non holistic approaches. The behavioral and neuroimaging studies [Sinha et al., 2006] further strengthen the motivation towards a non-holistic representation of faces. Initial methodologies selected only the areas of the face which are most invariant to expression (forehead + nose) [Mian et al., 2007] while later different combinations of explicitly defined face regions [Queirolo et al., 2010, Kakadiaris et al., 2007, Lin et al., 2007] were used.

In all cases the approach is to predefine interest zones instead of learning the salient zones from the data itself. This requires excellent pose normalization and furthermore, in most cases, does not include all the shape information within the explicitly selected regions. There are always subtle differences between faces and the transitions between the sub-sampled regions are as important as the information contained in the region [Sinha et al., 2006, result 4 and 14]. With enough data, all the variance in face shape can be described using sub-sampled regions of the face, provided appropriate descriptions are computed. Unlike previous research, the goal of the method described in this chapter is to take advantage of the inter-class similarities of faces and thus identify face regions in an unsupervised manner. Once these regions are identified and labeled, a Bagging [Breiman, 1996] inspired approach of an ensemble of classifiers is evaluated for classification.

3.1 Feature selection and extraction

The raw point data contained in each file has a variable number of points and as such, the surface shape information needs to be encoded in the form of a fixed sized vector, in order to facilitate optimum comparison operations. In the 3D literature the methods to compute the vectors are also called descriptors. Since the PCL library already contains implementations of feature descriptors for object recognition, it was worthwhile to evaluate the already implemented feature extraction algorithms.

According to the *no free lunch* theorem, there is no algorithm which performs best in all circumstances. By reviewing an experiment [Alexandre, 2012] on the accuracy and time performance of several 3D feature

Descriptor	Size	Harris3D	1cm grid	2cm grid	Avg %	Time (sec)	Rank
PFHRGB	250	77.89	79.32	79.75	78.99	5209.33	9.0
SHOTCOLOR	9+1344	69.2	75.53	73.42	72.72	325.00	6.0
PFH	125	48.42	56.75	55.49	53.55	2923.67	9.0
SHOT	9+352	42.53	55.49	46.84	48.29	121.00	**5.3**
FPFH	33	44.63	49.58	47.26	47.16	236.33	6.3
USC	variable	35.44	51.05	45.78	44.09	586.67	8.0
3DSC	variable	32.91	50.63	40.51	41.35	728.33	9.7
ESF	640	39.03	39.66	37.34	38.68	18.33	**5.3**
CVFH	308	20.63	35.23	19.41	25.09	13.67	**3.7**
RIFT	32	9.68	35.44	24.05	23.06	16.00	5.3
PPF	5	8.63	18.14	17.09	14.62	598.33	10.3
PCE	5	9.26	17.09	16.46	14.27	42.00	7.3
VFH	308	6.11	23.63	5.27	11.67	14.00	5.7

Table 3.1: The features evaluated in [Alexandre, 2012] are sorted according to the best average results. The second row is the size of the descriptor vector for one keypoint. The time is the duration of the whole experiment in seconds. The last row is $(2 \times Rank(accuracy) + Rank(speed))/3$ and represents a weighted average between two times the accuracy ranking and the time complexity.

descriptors implemented in PCL, three algorithms were selected (table 3.1 bold letters). The evaluation was performed on object and category recognition. For the experiment 48 objects belonging to 10 categories were used. The training set consisted of 946 point clouds while the test set contained 475 point clouds. An exhaustive search was performed through the complete training set to find the best match.

One of their important conclusions was that the recognition rates increase with the number of keypoints used, at the cost of processing time. Furthermore, color descriptors were demonstrated to have the best performance, yet this is intuitive for the purpose of object recognition where the texture information is specific for each object, as it can be observed from table 3.1. However, color information will not be further discussed here, since the texture information in faces is much more complex than in objects (e.g. red mug, blue mug).

The pipeline used for the experiments extracts features only from a subset of the original data in order to speed up processing. Two methodologies are used: the Harris3D combined corner and edge detection algorithm [Harris and Stephens, 1988] which uses the Hessian matrix of intensities in order to extract keypoints, and a simpler method of uniformly sub-sampling the input data (two different grid sizes) to use as keypoints. Each object is represented as a matrix of histograms, each vector containing the output of the feature extraction algorithms. For each algorithm, one matrix is extracted and compared against a database containing the same type of precomputed representations. The matching is done using the l_2 distance (eq. 3.1) as a sum of the centroids and standard deviations computed for each dimension of the object A and the target B (eq. 3.2).

$$l_2(A,\ B) = \|A - B\|_2 = \sqrt{\sum_{i=1}^{n} |a_i - b_i|^2} \qquad (3.1) \qquad d(A,\ B) = l_2(c_A,\ c_B) + l_2(std_A,\ std_B) \qquad (3.2)$$

The performance of 13 different feature extraction methods as compared in [Alexandre, 2012] is presented in a form that considers object recognition and time complexity in table 3.1. Out of these, three were selected on the basis of sufficient vector size and a good balance between accuracy and speed. A rank was given to each descriptor and computed as a weighted average between the rankings of each algorithm in accuracy and speed: $(2 \times Rank(accuracy) + Rank(speed))/3$. The lowest numbers represent the best ranks and are marked with bold in the last column.

Since objects from the same category can be very similar (such as faces), only the object recognition performance results were considered here, although the category recognition accuracy results were reported as well — and were indeed much higher. The results are displayed in table 3.1 with the feature descriptors sorted in descending order of average accuracy. First of all, the color descriptors were completely ignored. FPFH, USC, 3DSC, RIFT, PPF and PCE were not considered as suitable candidates since the size of the histograms is small and this suggests a weak discriminative power. The remaining descriptors according to rankings in descending order of performance are: CVFH, ESF, SHOT, VFH, PFH. CVFH and VFH were also eliminated since they remove the points with high curvature, followed by a smooth region growing algorithm which would eliminate much of the details necessary for encoding the face shape details. We are therefore left with three descriptors which are discussed in further detail in the following paragraphs.

3.1.1 SHOT

The Signature of Histograms of Orientations (SHOT) [Tombari et al., 2010] is based on computing a robust local reference frame using eigenvalue decomposition around a keypoint. A spherical grid (fig. 3.1) is then centered on the same point and for each bin in the grid a weighted histogram of normals is computed according to a function of the angle between the normal at each point within the corresponding part of the grid bin and the normal at the keypoint.

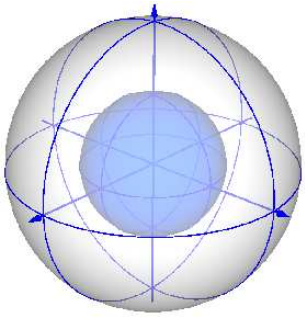


Figure 3.1: Signature structure sketch of SHOT. From [Tombari et al., 2010]

The results are concatenated, the first 9 values represent the reference frame followed by 11 shape bins times the 32 bins — resulting from 8 azimuth divisions, 2 elevation divisions and 2 radial divisions of the spherical grid, with a total number of 352 values. To achieve robustness to variations of the point density, the whole descriptor is normalized to sum up to 1.

3.1.2 PFH

Point Feature Histograms (PFH) [Rusu et al., 2008] generalize the mean curvature around the selected keypoint by encoding the geometrical neighborhood using a multi-dimensional histogram of 125 values. The surface variations are sampled into a hyperspace by taking into account *all* the interactions of the directions of normals of the neighbors for one keypoint fig. 3.2.

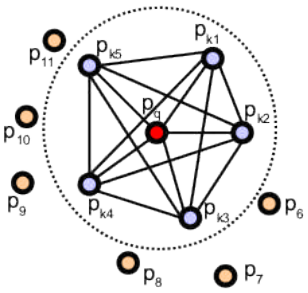


Figure 3.2: PFH: Selection of neighboring points.

3.1.3 ESF

The Ensemble of Shape Functions (ESF) [Wohlkinger and Vincze, 2011] is a global descriptor and does not require the computation of normals, hence its low computational complexity. It consists of 10 concatenated histograms — three angle, three area, three distance and one distance ratio shape functions with 64 bins each, hence a final dimensionality of 640 values. The point cloud is first approximated into a $64 \times 64 \times 64$ grid. This might suggest that it is not a good descriptor, however we will be dealing with *patches* of surfaces, hence the approximation will not have such dramatic consequences since we are already dealing with a sub-sampled region of the face. The first three histograms (fig. 3.3) are computed using distinct shape functions describing distance, angle and area distributions on the surface of the point cloud. Three types of measurements are made based on points ON the surface, OFF the surface

and MIXED. From the mixed type, a fourth histogram is computed as the ratio of the line which resides on and off the surface.

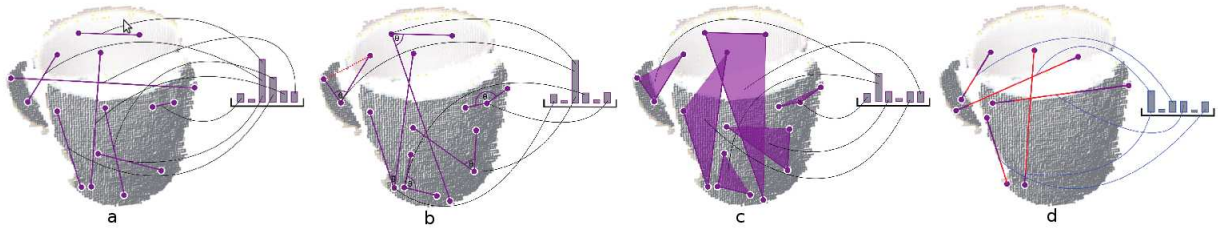


Figure 3.3: Points randomly selected from the point cloud are used to form four types of histograms (left to right): a) the distance between random points is computed; b) the angle between the lines; c) area described by three points; d) ratio of the lines which are both on and off the surface. From [Wohlkinger and Vincze, 2011]

3.2 Unsupervised face region labeling

Since the goal of this experiment is to test the clustering performance of face regions based on three types of features, eight classes ($\Omega = 8$) from the "roll" set which were accurately normalized were selected in order to eliminate the pose variance. Since the original frames were normalized, there was no requirement for the descriptors to be rotation invariant. Before the division of the original frames into patches, this small dataset $\mathcal{D}_s$ consisted of 152 sensor observations $\mathcal{S}$. The subdivision according to the grid size is listed in table 3.2 on page 29.

By using different grid sizes, we were able to focus on the evaluation of the discriminative capacity of the features according to the size of the sub-sampled regions. The frames used in this experiment do not contain observations with large zones of missing data, since for example a face which is tilted 45 degrees will only have a few missing pixels from the forehead / chin. Hence, we can consider this eight class dataset to be a slightly more difficult version of a purely frontal set with no pose or expression variance. An illustration of the classes used in this experiment is given in fig. 3.5.

Figure 3.4: The grid (3cm here) is projected orthogonally onto one pose normalized face. Each patch is colored differently, for visualization purposes. The patches colored blue are discarded since the number of pixels in those patches is smaller than the average number of pixels per patch.

3.2.1 From grids to patches

As outlined earlier, holistic methods for face recognition are not robust, since faces are too dynamic to be described in such a way. As such, a new piecewise strategy is evaluated, where each sensor observation $\mathcal{S}$ is divided into separate regions using a fixed grid. Since an appropriate value for the resolution of the grid is not yet known, multiple grid sizes are evaluated, namely 2, 3, 4 and 5 cm cell sizes for patches. The grid is projected orthogonally onto the face (fig. 3.4), which is then broken down into separate patches. This is followed by a feature extraction step using the three descriptors discussed in the previous section.

Thus, three new feature datasets are formed with feature vectors $\vec{V}$. The same process is repeated for all the observations in the current dataset $\mathcal{D}_s$ with the eight classes Ω depicted in fig. 3.5. After the division in patches, we are left with patches which have class labels but do not yet have "face region" labels. For the labeling of clusters of face regions using the sub-sampled patches two methods are used: Self Organizing Maps and K-means. Uniformly distributed random region assignment was also tested to validate clustering, however the results are not reported.

Figure 3.5: The 8 classes used in this experiment. For each class, the frames are overlapped for each class and are slightly rotated towards the right to show the 3D nature of the observations. In the center, all the data is overlapped, showing that all the camera observations have approximately the same canonical position in 3D space.

3.2.2 Self-organizing maps

Inspired from the sensor-motor mappings of the mammal brain, which also automatically organizes information topologically, self-organizing maps (SOM) [Kohonen, 2001] have the advantage of providing a good way of obtaining a qualitative assessment of the topological relationships between clusters, in addition of a qualitative evaluation of the clustering itself. Belonging to the family of vector quantization methods, SOMs permit the representation of multidimensional data in much lower dimensional spaces since they create a network that stores information where the topological relationships within the training set are maintained. Hence, SOMs can be used to both cluster data and reduce dimensionality. Another motivation for using SOMs is that they have the tendency of assigning the same number of instances to each class.

The self-organizing map is a network of E neurons which is fully connected to the input layer. Each neuron has a unique topological position — in our case two dimensions SOM_X and SOM_Y — and has a vector of weights $\vec{W}$ of the same dimension n as the input vector $\vec{V}$. There are no connections between the neurons themselves, only between inputs and neurons.

Since we are describing a surface, we will be using a two dimensional SOM. Different sizes of the map (E, the number of clusters) will be evaluated according to the average number of patches per observation. Due to missing data, the number of patches per observation $\mathcal{O}$ will not be constant. This is intuitive since each grid size divides the face into a different number of patches. The steps involved in the online training are described below in detail:

0) The training is initialized by randomly assigning n weights for each neuron in the network. As such the final topology of the map is directly influenced by this step. The algorithm stops once the maximum number of iterations T is reached. The weight values $\vec{W} \in [0, 1]$.

1) A randomly selected input vector $\vec{V}$ is then compared to all the weight vectors $\vec{W}$ in the SOM matrix $SOM_{X,Y}$ using a distance function d, in our case the l_2 (eq. 3.1). Since the vector selection is stochastic, the training procedure does not guarantee identical results on the same dataset. The neuron with the smallest distance d, commonly known in the literature as the Best Matching Unit (BMU) — eq. 3.3 — is then selected as the "active" neuron.

$$BMU = \arg\min_{X,Y} d(\vec{V},\ SOM_{X,Y}) \tag{3.3}$$

2) After the BMU is found, its neighborhood is selected according to a radius r. Different distance functions can be used, since the data is in an actual grid, the l_1 norm (eq. 3.4) was used in this case. Initially the radius is set to cover the whole map, however it is decreased exponentially each time-step t according to eq. 3.5, where N is the total number of observations. The decay could also be modeled as a linear function, however for the online version this guarantees that the topology will settle early and will generate a uniform distribution of the data between clusters later on.

$$l_1(A,\ B) = \|A - B\|_1 = \sum_{i=1}^{n} |a_i - b_i| \tag{3.4}$$

$$r(t) = r * \exp\left(-\frac{t}{N}\right) \text{ where } t = 1, 2, \ldots, N \tag{3.5}$$

3) This is the step that contains the "core" idea behind SOMs. The weights of all the neurons W_ε in the "current" neighborhood are adjusted according to a learning rate η which also decreases exponentially

with time as in eq. 3.5. In addition, and more importantly, in order to generate an appropriate topology we intend to adjust the weights of each neuron (eq. 3.6) also as a function of the vector's distance from the BMU. This is usually done using a Gaussian radial basis function (RBF) ϕ — eq. 3.7 — which is a function of time since it also depends on the radius.

$$\vec{W}(t+1) = \vec{W}(t) + \eta\,\phi(t)\,(\vec{V}(t) - \vec{W}(t)) \tag{3.6}$$

$$\phi(t) = \exp - \frac{l_2(BMU,\, W_\varepsilon)^2}{2r^2(t)} \tag{3.7}$$

4) The process is repeated from step **1)** until the maximum number of iterations T is reached. Previously unseen input vectors presented to the network will then activate the neurons in the zone with similar weight vectors. This is useful since the training can continue even if new information is added.

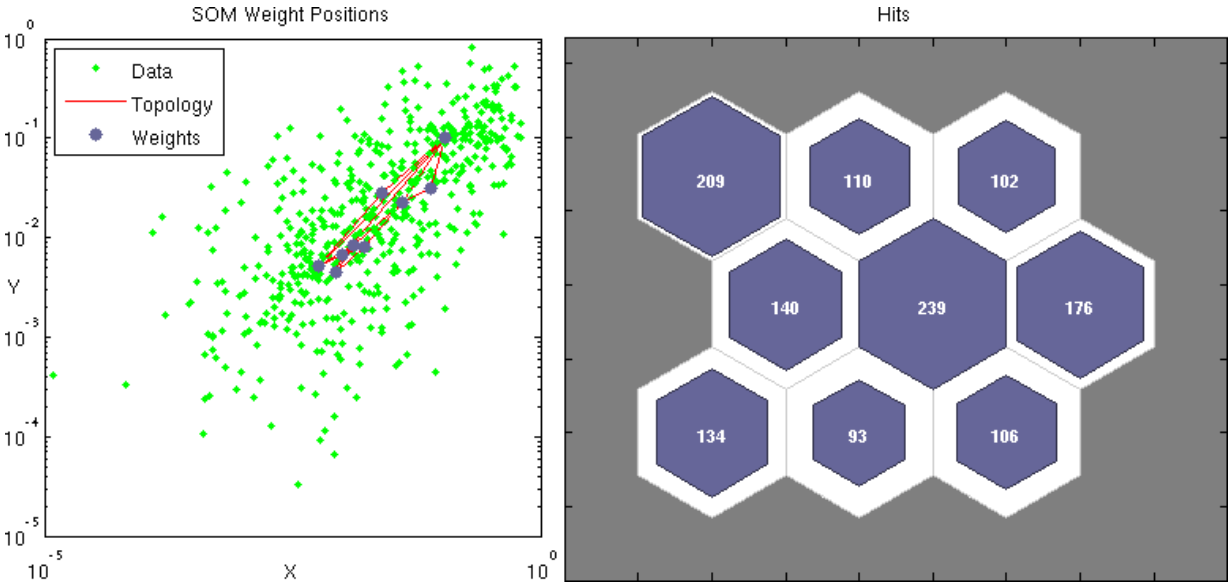


Figure 3.6: Left: The distribution of the weights and the topology on the projected 2D space of a SOM using 5cm SHOT descriptors, logarithmic scales. It can be observed that there are no obvious agglomerations of the data points on this projected 2D space. This could also be an indication that the features might not be descriptive enough of the shape — the patches might be too similar. Another visualization technique might, however reveal more towards clustering performance. Right: The corresponding 3 by 3 hexagonal grid, each neuron can have a maximum number of six connections to adjacent neurons, hence it has the potential to generate a better topological distribution of the input space. The number in each hexagon is the number of hits per cluster, the distribution of the input space.

If input vectors appear with approximately equal probability, then the neurons of the self-organizing map will order themselves with approximately equal distances between themselves. Otherwise, the feature map will allocate neurons to an area in proportion to the frequency of input vectors. This means that the SOMs not only learn to categorize the input but also learn the topology and thus reflect the distribution of the input in the number of instances per cluster, as can be observed in fig. 3.6 right side. As with all algorithms the choice of the decay function (eq. 3.5) and the learning rate η directly affects the topology and the distribution of the input space.

Although the online version of the training was explained here, in the performed experiments the batch version was used since it is much faster. The main difference between the online and the batch version is that the topology and distribution learning stages are done in two large separate steps in batch mode instead of alternating between the two as in online mode. The batch training method was set to run for $T = 100$ epochs. The first step is topology learning, where $r = 3, \eta = 0.9$ and all the vectors in the input space are presented until the radius becomes one. During this step the neuron weights arrange themselves according to the input space, thus the topological order is found. In the next training step T is set to 100 again and the radius becomes less than one and $\eta = 0.2$, which implies that only the weights for the BMU are adjusted, thus the distribution of the data is learned.

3.2.3 K-means

The goal of K-means is to divide a total of N examples into E sets with the objective of minimizing the average squared l_2 distance (3.1) of the feature vectors $\vec{V}$ from their cluster centers $\vec{\mu}$. The cluster center or centroid is defined as the mean of the vectors in a cluster $\varepsilon \in [1, E]$ (eq. 3.8).

$$\vec{\mu}_\varepsilon = \frac{1}{|\varepsilon|} \sum_{\vec{v} \in \varepsilon} \vec{v} \tag{3.8}$$

It is common that the input vectors are length-normalized $\|\vec{V}\|_2 = 1$ or standardized. The optimum clustering result would be the situation where there would be no overlap between clusters and all clusters would have the centroid coincide with the center of gravity of the sphere described by the vectors in that cluster. Each centroid is thus an average representative of the collection of all observations within that cluster. The representation quality can be computed by measuring the sum of squared distances from each vector in the cluster to its centroid (eq. 3.10), formally known in the literature as the residual sum of squares (RSS). Then, the objective is to minimize the total RSS (eq. 3.9) and since the vectors have the same dimension n, this is indeed equivalent to minimizing the average squared distance.

$$minimize\ RSS = \sum_{\varepsilon=1}^{M} RSS(\mu_\varepsilon) \tag{3.9}$$

$$RSS(\mu_\varepsilon) = \sum_{\vec{v} \in \varepsilon} |\vec{v} - \vec{\mu_\varepsilon}|^2 \tag{3.10}$$

The algorithm alternates between two associate-move steps. The training process is described below:

0) As we have also observed in SOMs, the final result is dependent on the initialization of the algorithm. In this case, the sensitivity due to initialization is even higher and it frequently happens that suboptimal partitions are found. The cluster centroids $\vec{\mu}$ are initially called seeds and are typically initialized randomly as one of the dataset vectors. One heuristic to improve the *seeding* is to make sure that the distances between the seeds are as large as possible. It is quite common, for small values of E, to *reseed* and keep the result with the lowest RSS. In our case, the seeding was repeated 10 times.

1) Association: The distance between all the input vectors to each centroid is computed and the vectors get assigned to the closest centroid (eq. 3.11). In the online version, the procedure is to select one example at a time, rather than all at once. This step can be compared to the topology mapping step in SOMs and with the Expectation step in the Expectation-Maximization (EM) algorithm [Moon, 1996].

$$\vec{V}_\varepsilon = \arg\min_{\varepsilon} l_2(\vec{v},\ \mu_\varepsilon) \tag{3.11}$$

2) Relocation: After all the observations are allocated to a centroid, the centroids themselves are moved (eq. 3.8) to the new means. In the online version (eq. 3.12) since one observation is selected at a time, a

learning rate η is used to move the centroid only slightly towards the new mean μ of the cluster instead of replacing it completely. Then, each μ_ε becomes a weighted average of the observations that were closest. Usually $\eta \in (0,1)$ is set as a constant, however the number of current observations for each cluster can also be used. This step is reminiscent of the distribution learning step in SOMs and the Maximization step in the EM algorithm. It can sometimes happen that there are no observations closest to μ_ε in which case, most commonly E is decreased and the centroid is removed. The same strategy was used in the experiments here.

$$\vec{\mu}_\varepsilon(t+1) = \vec{\mu}_\varepsilon(t) + \eta \, |\vec{v} - \mu_\varepsilon|^2 \text{ where } \eta = \frac{1}{|\mu_\varepsilon(t)|} \tag{3.12}$$

3) The algorithm then alternates between **1)** and **2)** until the positions of the centroids $\vec{\mu}_\varepsilon$ do not change anymore, which was also the case in the current experiments. Another stopping condition can be reaching an RSS below a threshold value provided the maximum number of iterations has not been exhausted, since the threshold might be never reached.

3.2.4 Grid projection datasets

Four datasets were obtained for each type of feature, according to the size of the grid. Regardless of the type of feature used, the number of subdivisions was the same. Since the grid divides each frame into patches, the number of observations increases when the grid size becomes smaller. Thus, in total there are 3 descriptors $\times$ 4 grid sizes = 12 datasets which were generated in order to evaluate the performance of each type of feature according to grid size.

Since the patches came from different regions of the face their position in space was normalized before the feature extraction step. Thus, each patch was translated in such a way that the origin of the three axis coincided with the center of mass of each patch as can be observed in fig. 3.7.

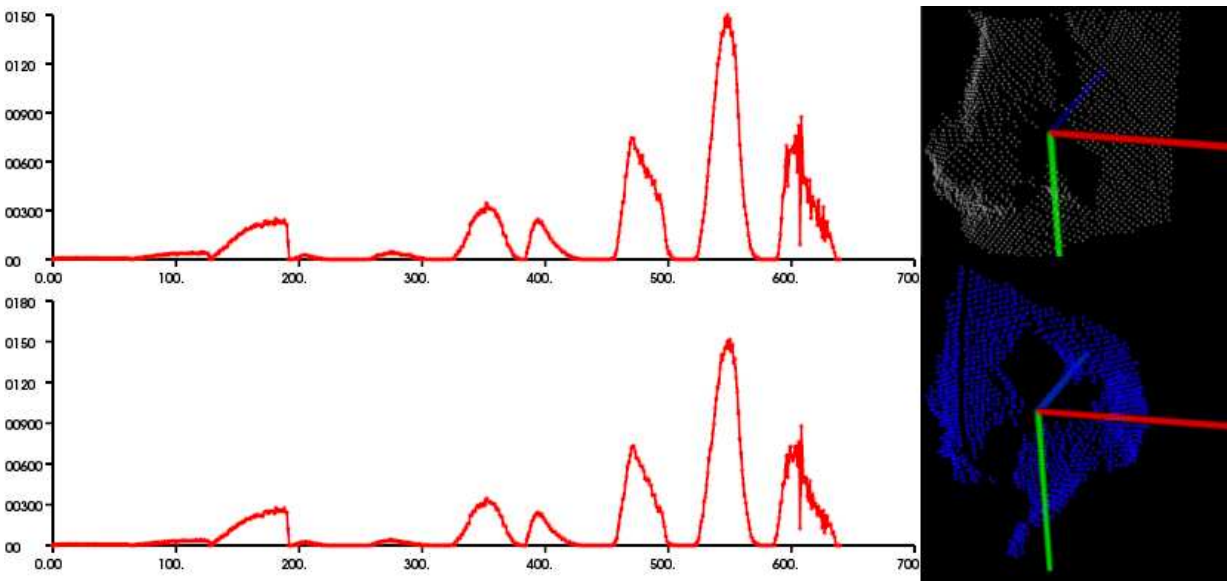


Figure 3.7: An ESF histogram of a nose patch which is rotated around its Cartesian centroid. One can observe that the whole histogram is composed of smaller concatenated histograms. The two histograms differ slightly, which is an indication that the descriptor is not entirely rotation invariant.

The number of patches per class, according to the division of the grid is shown in table 3.2. For clarity purposes, from this point on, the patches will be referred to as observations, instead of the camera frames, since the face region patches are used to extract features and subsequently to train separate classifiers. It can be seen that the number of frames — *sensor observations* — is not equal for each class. This implies that the accuracy will be biased by the non-uniformity of observations per class. The total number of observations and the average number of observations per frame (complete face) is also listed. The average number of observations per frame $\hat{E}$ — the last line in the table — is a good indication towards the number of clusters E that should be chosen, assuming that there is an even distribution of patches per face region.

Class	Frames	5cm	4cm	3cm	2cm
amir_s	17	143	205	368	774
ayla_k	13	108	149	250	565
ben_w	30	278	372	644	1402
davey_s	20	162	244	436	943
diederik_v	9	80	111	188	408
florin_s	31	242	351	633	1349
joost_b	13	109	148	275	581
marco_w	19	187	270	460	975
Total	152	1309	1850	3254	6997
$\hat{E}$	-	8.61	12.17	21.41	46.03

Table 3.2: Number of observations N per grid division dataset. The average number of patches per frame $\hat{E}$ is a good indication of the ideal number of clusters E.

3.3 Classification

The problem of classification has been extensively studied in machine learning. The most basic type of classification is binary, where a decision has to be made whether an observation belongs to a class label ω or not. Conversely, multi-class problems involve more complicated algorithms which try to reduce the problem to the binary case or try to fit a single function which would separate each class from the rest. Two popular algorithms are SVMs [Cortes and Vapnik, 1995] and Multi-Layer Perceptrons (MLPs), a type of Feed-forward Neural Networks, both having advantages and disadvantages.

3.3.1 Multi-class paradigms

In this type of classification, each training example $\vec{V}$ belongs to one of Ω different classes and the goal is to estimate a function f which, given a new query observation, will correctly predict the class label of the probe. There are two main paradigms for multi-class problems. As mentioned in the previous chapters, face recognition can be of two types: *face verification* and *face identification*. This is also a direct consequence of the classification paradigm. The former type is linked to "All-vs-All" (AVA) classification, while the latter corresponds to the "One-vs-All" (OVA) paradigm. The typical classification algorithm that is widely used in the face recognition literature, is SVMs [Cortes and Vapnik, 1995], mostly due to their popularity in recent years.

In the "All-vs-All" case, also called "One-vs-One", $C(C-1)/2$ binary classifiers are built to distinguish between all the possible pairs of classes i and j. The decision can then be based on the classifier with the

highest confidence. If we take f_{ij} to be the classifier which has class i as positive examples and class j as negative examples then one can classify using eq. 3.13.

$$f(x) = \arg\max_i \left(\sum_j f_{ij}(x) \right) \text{ where } j = 1, 2, \ldots, C \tag{3.13}$$

The most common classification scheme, the "One-vs-All" implies training N different binary classifiers. For each class label i, a classifier is trained using the observations of i as the positive examples, while everything else $\neg i$ is used as negative training examples. The decision is then made by choosing the prediction with the highest confidence score as in eq. 3.14.

$$f(x) = \arg\max_i f_i(x) \text{ where } i = 1, 2, \ldots, C \tag{3.14}$$

The difference between these two multi-class paradigms is mainly computational. Intuitively All-vs-All requires $\mathcal{O}(N^2)$ classifiers instead of $\mathcal{O}(N)$, however each classifier is on average much smaller. The time required to train a classifier is a good indication towards a choice between the two methodologies. When the training time is super-linear in the number of examples, the former is a better choice. SVMs are a typical choice for this type of classification scheme. Another advantage is the easiness to parallelize the training procedure, which has the potential to further speedup processing. However, if the training of one classifier from a "One-vs-All" can be performed over the entire data set using a matrix factorization, then the speed could increase at the expense of using more memory.

While these are the classic paradigms which try to reduce multi-class problems to binary, there have been several other approaches which aim to find one solution for the complete dataset. In [Hsu et al., 2003] the authors conclude that AVA methods are more suitable for practical use than the other methods, after they empirically compare the performance of SVMs on all the proposed methodologies.

3.3.2 Decision boundaries in supervised learning

The goal of margin classifiers is to find a function f (hyperplane) which best describes a decision boundary in order to separate the input space as accurately as possible. The distance of a query vector $\vec{V}$ from the separating hyperplane is the margin corresponding to the query. Decision boundaries do not always have to be clear. For a binary classifier, the transition from one class to the other in feature space might be gradual. Hence, membership in one class or another might be ambiguous.

Multi-Layer Perceptrons (MLPs) are feed-forward artificial neural networks which consist of three or more layers in a graph with neurons (nodes) which correspond to non-linear transfer functions. Usually the logistic sigmoid $f(v) = 1/(1 + \exp(-v))$ with $f(v) \in [0, 1]$ or the hyperbolic tangent $f(v) = tanh(v)$ with $f(v) \in [-1, 1]$ functions are used. One node i in one layer is connected through weight vectors $\vec{W}_i$ with all the nodes in the next layer. These functions are used since they model the firing of biological neurons in the brain and are differentiable — which is a requirement since the derivatives of the activations need to be computed during back-propagation [Rumelhart et al., 1986], the classical choice for training. The combination of weights and the non-linear functions are the main difference between simple perceptrons [Rosenblatt, 1958] and MLPs, which allows non-linear input-output mappings to be learned. The advantage with neural networks is that they are more biologically plausible and they require only a few parameters to be tweaked, the learning rate η and the number of hidden units and/or layers. Furthermore, in online training mode, they do not require as much memory as for the computation of the kernel matrix in SVMs. The complexity of the decision boundary in MLP is directly affected by the number of

hidden layers. With no hidden layers, an MLP can only learn linear functions, while with one hidden layer it can learn generic functions composed of convex and concave decision boundaries. If the number of layers is greater than two, then the network can learn even more complex functions.

In the case of using MLPs on muli-class problems, a typical scalability problem arises. For a finite number of classes, we could for example assign one output node to each class, then the problem would be trivial. However, when introducing a new class, a new output node is required. This implies that the network should be retrained, perhaps on the whole dataset since the weights and the gradients would need to change to accomodate for the new class. Depending on the complexity — the number of hidden layers it is therefore necessary to re-train to compensate for the newly added output neuron. For large networks, the gradient will fade and it might be necessary to re-train using the whole dataset. This is quite impractical. However, the other option is to use a single output and multiple MLPs with the AVA approach.

Support Vector Machines (SVMs) are maximum-margin classifiers which use the "kernel trick" to project the input data to higher dimensions in order to find an appropriate decision boundary. SVMs construct hyperplanes according to some kernel function which can be polynomial, RBF (eq. 3.7), or sigmoidal. They are highly sensitive to the training parameters, which require specific tweaking according to the kernel type. Nevertheless, there are numerous ways in which to explore the parameter space such as grid search followed by fine tuning using particle swarm optimization (PSO) [Eberhart and Kennedy, 1995]. However, SVMs are simpler as an architecture and thus their behavior can be analytically understood much easier.

Although SVMs and MLPs might seem quite different, in [Collobert and Bengio, 2004] the authors show that under various simplifications (for example threshold-based "hard" transfer functions) "MLPs are in fact SVMs which are maximizing the margin in the hidden layer space, and where all hidden units are also SVMs in the input space.". They also propose a new method to train MLPs by first making a parallel between SVMs and Perceptrons [Rosenblatt, 1958] in order to suggest a cost function which is based on the conclusion that "Under certain simple conditions, a perceptron is equivalent to an SVM". Then they extend this idea to the non-linear case by considering threshold-based neurons as SVMs.

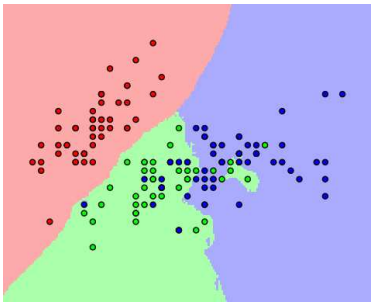


Figure 3.8: The decision boundary of KNNs on the widely used Iris recognition dataset with $\Omega = 3$ classes, $|\vec{V}| = 4$ features and $K = 15$ neighbors. The weights $W = 1/\mathrm{d}$ where d is the l_2 distance to a neighbor.

The main difference between MLPs and SVMs is that the former aim to learn the decision boundary which minimizes the empirical error (typically MSE), while the latter try to learn the decision boundary which gives the best generalization. In [Collobert and Bengio, 2004] the authors start from this difference to link the two classification algorithms by optimizing generalization performance for MLPs.

The K-Nearest Neighbor (KNN) algorithm has very strong consistency results which have been analytically proven [Cover and Hart, 1967]. The scalability is evident, since as the number of examples N approach infinity, the algorithm is guaranteed to have an error which can not be worse than twice the Bayes error rate. Furthermore, as the number of examples N increase, KNNs are guaranteed to approach the Bayes error rate, according to some value of K. The complexity of the decision boundary is therefore directly a function of the number of neighbors K (fig. 3.8). The larger K is, the smoother the classification boundary. Thus, we can say that the complexity of the decision boundary is lower when K increases. However, lower K *might* imply poor generalization, depending on the feature space. There is always a balance between K and the error rate, which is the optimization objective. The Bayes error rate is an indicator of the lower bound of the error according to the distribution of the training data. The main problem with KNNs is the "curse of dimensionality", especially when using the euclidean distance with a large number of dimensions n.

3.3.3 Relative performance analysis

Since our goal in this chapter is to compare the performance of each type of feature per grid size, it was faster to use a fast classification algorithm which will provide insight towards the *relative* performance among the feature descriptors within the twelve datasets.

K-Nearest Neighbor (KNN) is a "supervised" non-parametric instance-based learning algorithm for classification used mainly because of its simplicity and scalability. This classifier is commonly based on the Euclidean distance (eq. 3.4) between a query and a training set, with the classification rule based on assigning to the query the label of the majority vote of its K nearest training samples. In practice, K is usually chosen to be odd in order to avoid ties. If $K = 1$ then the instance is assigned to the class of its closest neighbor, a special case called Nearest Neighbor classification.

It is common to use weights $\vec{W}$ during the voting procedure, such that the closer the neighbor, the higher the contribution towards the average final vote. While this makes the decision boundary fuzzy, the weighting scheme is a way of unbiasing the classifier in cases where the number of examples for a particular class outnumber the others. In such cases, one region of the feature space could be more dense with examples from a particular class, in which case the weights penalize the distance to the new sample.

The K-Nearest Neighbor algorithm is sensitive to the local structure of the data, hence the distance function is very important. In the case of high dimensional vectors, the Euclidean distance (eq. 3.1) becomes irrelevant since the distance to all neighboring points can be almost identical [Cover and Hart, 1967]. Furthermore, the Euclidean distance can not distinguish between correlated variables. In the case where the vector contains two features which are identical, it can not take into account that the duplicate does not bring any new information and will therefore bias the results more heavily based on the redundant feature. Hence, dimensionality reduction using methodologies such as PCA [Jolliffe, 2005] is a normal regularization step when using this distance function.

The Mahalanobis distance (eq. 3.15) is a scale invariant dissimilarity measure and takes into account the covariance among the features when calculating distances, thus the problems of scale and correlation are no longer an issue. For example, on two dimensional feature spaces ($|\vec{V}| = 2$) the Euclidean distance represents the probability distribution evenly by a circle, if we imagine a X vs Y plot of the feature vector. The Mahalanobis distance is the distance of the test example from the center of mass divided by the width of the ellipsoid in the direction of the query vector.

$$d_M(\vec{A},\ \vec{B}) = \sqrt{(\vec{A} - \vec{B})^T \Sigma^{-1} (\vec{A} - \vec{B})} \quad (3.15) \qquad \Sigma = [\sigma_{jk}] = \frac{1}{N-1} \sum_{i=1}^{N} (x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k)) \quad (3.16)$$

However, the Mahalanobis distance can be used only if the sample covariance matrix (eq. 3.16 where $\bar{x}$ = mean of x) of the dataset is positive semi-definite. Furthermore, the covariance matrix should be re-computed each time new examples are added. The covariance matrix can be manipulated in order to achieve the desired results, however this would also lead to errors. Typically PCA is used for dimensionality reduction in combination with the Euclidean distance. There are other methods for dimensionality reduction such as a brute force search through removing or adding one feature at a time and comparing cross-validation results on any form of classification scheme, which in some cases is most useful. In our case, neither methods have resulted in sparse representations with acceptable discriminative power on these datasets.

3.3.4 Ensemble learning

In ensemble learning, multiple experts are strategically combined to solve computational tasks such as function approximation or prediction. The effectiveness of such learning paradigms comes from the diversity of the individual experts. This diversity can reside in the type of classification algorithm, the variance in parameters used during training of the same type of classifiers, in the type of features used for each classifier or using subsets of / re-sampling the training data (Bagging [Breiman, 1996] or Boosting [Schapire, 1990]). An exhaustive review on ensemble diversity matters is provided in [Brown et al., 2005].

Moreover, a crude taxonomy of ensemble algorithms can come from the topology of the set of experts, which can be either in "series" or in "parallel" and can be supervised by meta-classifiers as in the Mixtures of Experts method [Jacobs et al., 1991] or Stacking [Wolpert, 1992]. In other words, the data can come at the same time to all experts, or is sequentially filtered in subsets and passed from one expert to the other. It is quite evident already that the possibilities are numerous and usually depend on the type of problem. By combining the decision boundary of several such experts, it is possible, although not guaranteed, that the resulting learning algorithm will perform better than one single classifier. Another intuition behind ensemble learning is that if individual classifiers make different errors on different instances, then the combination will most likely reduce the overall error.

Additionally, there is the possibility with ensemble learning to have a meta-classifier which supervises the decisions of each classifier. One strategy is known in the literature as Stacked generalization [Wolpert, 1992]. By training the "critic" at the same time with the experts, the importance of each classifier can be changed according to class or type of variance in input data (camera observations with missing data or expressions). As such we are in fact learning a weight space over the whole dataset. Mixture of experts [Jacobs et al., 1991] also make use of meta-classifiers to learn the ideal weights for gating the decisions of several classifiers.

In the following experiment the diversity comes from using different subsets of the input data (patches) which, provided the convergence of the clustering algorithm, should ideally represent different face regions. Thus, the core idea resides in having specialized classifiers for each region of the face. Since the performance of each feature is unknown nor the optimum scale, the following experiment is primarily focused on identifying the capabilities of each feature type. A favorable consequence of using subsets of training data is that we are less likely to require complex decision boundaries for the classifiers when increasing the number of examples N and the number of classes Ω. Hence, the decision boundary that separates the dataset may be too complex, or lie outside beyond the set of functions that can be described by one single classifier.

Since we want to observe the performance and impact of clustering and also compare the results of the ensemble, one single expert was trained using the same data. We call this one single classifier an expert since it makes use of either the majority voting scheme or sum rule when making decisions. However, it should be clear that this expert does not take into consideration the face region labels obtained in the clustering step. Thus, to conclude we are using a set of experts in "parallel" which are trained in a similar way to bagging (although without replacement) in order make a decision about the class of the *probe* sensor frame.

While the established algorithms have their intrinsic combination rules, an ensemble of classifiers can be simply trained on different subsets of the training data. Such is the case in the following experiment where two different combination rules were succinctly evaluated. There are two major ways of combining classifiers: algebraic, which operate on the continuous space of the direct output of individual experts or voting methods which operate on the final decision labels returned by each expert. The problem of combining classifiers has been studied in [Kittler et al., 1998] where it has been shown in a comparative study that the sum rule outperforms other combination schemes since it is most resilient to error.

3.3.5 Majority voting

In [Kittler and Alkoot, 2003], a subsequent study on ensemble learning using subsets was performed and it was empirically demonstrated that for a number of experts smaller than five, majority voting is slightly better, however, when there is a large number of experts, the sum rule outperforms voting.

In the case of abstract labels, the decision of classifier C_ε on observaton v can be defined as $\phi_\varepsilon^\omega(v) \in \{0,1\}$, $\varepsilon = 1,\ldots,E$ and $\omega = 1,\ldots,\Omega$ with E classifiers and Ω classes, while in the case of continuous outputs, the decision is continuous as in $\phi_\varepsilon^\omega(v) \in [0,1]$ with the outputs of all the classifiers length-normalized $\|\vec{\phi}\|_2 = 1$.

Voting based methods operate on labels only and thus the final vote is simply the class with the largest number of accumulated votes from each classifier. Weights w can be added to each classifier according to some objective function.

$$\Omega(v) = \arg\max_{\omega} \sum_{\varepsilon=1}^{E} w_\varepsilon \, \phi_\varepsilon^\omega(v) \tag{3.17}$$

3.3.6 Posterior probabilities of KNN experts and the sum rule

Since the features correspond to specific areas of faces, in a similar way to bagging (although without replacement) the feature dataset $\mathcal{D}$ is partitioned in E distinct sets $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_E\}$ obtained during clustering.

Even though the subsets $\mathcal{D}_{1\ldots E}$ are distinct — in order to generate diversity between experts — they are correlated by keeping track of the original camera observation $\mathcal{O}$ from where the features were extracted. Thus, from the complete dataset $\mathcal{D}$ with N examples, any camera observation $\mathcal{O}^i = \{\mathcal{O}_1^i \in \mathcal{D}_1, \ldots, \mathcal{O}_E^i \in \mathcal{D}_E\}$ can be composed of a variable number of feature subsets. This happens since the clustering is not optimal and we can thus observe multiple hits from one particular region per camera observation.

For each of the subsets $\mathcal{D}_\varepsilon$ with $\varepsilon \in [1,E]$, one expert C_ε is then trained. During testing, for any query frame, each expert can thus process more than one feature vector and thus return multiple results. In this case, the class posterior probability estimates $\vec{P}_\varepsilon$ are simply summed for each expert. In this way, we make sure that no data is discarded. As such, each camera observation $\mathcal{O}^i$ is selected and used as probe. Then, for each subset $\mathcal{D}_\varepsilon^i$ the distance to each centroid is computed and the corresponding experts C_ε are activated for processing the subsets.

Then, each expert C_ε returns a vector $P_\varepsilon(v) = \{P_\varepsilon(\omega_1|v), \ldots, P_\varepsilon(\omega_\Omega|v)\}$ with $v \in \mathcal{D}_\varepsilon$ which contains the posterior probability of the expert C_ε for each class Ω. The posterior probability that a query feature vector v belongs to class Ω is computed using equation 3.18 where $w_\Omega(k)$ is a weight vector computed as the inverse square distance $1/d^2$ where d is the distance between the query vector v and a neighboring vector.

$$P_\varepsilon(\omega|v) = \frac{\sum\limits_{k\in\hat{\mathcal{K}}} w_\Omega(k)\,\Phi(v,k)}{\sum\limits_{k\in\hat{\mathcal{K}}} w_\Omega(k)} \text{ where } \hat{\mathcal{K}}(v) = \text{the closest neighbors} \tag{3.18}$$

$$\Phi(v,k) = \begin{cases} 1 & \text{if } k \in \hat{\mathcal{K}}(v) \\ 0 & \text{otherwise} \end{cases} \tag{3.19}$$

Here, the final ensemble decision is the class Ω that receives the largest support after the algebraic expression is applied to the individual supports obtained by each class. Essentially the ϕ in equation 3.17 is replaced with P_ε which contains the posterior probabilities for each class, obtaining eq. 3.20. Single layer perceptrons with sigmoid activation functions could be used to learn the weights. In [Lin et al., 2007] LDA was used to compute the weights. However, in this chapter the weight vectors for each expert $\vec{w}_\varepsilon = 1$, hence each expert has equal importance towards the final decision. These weights could also be adjusted according to the topological relationship between the clusters and the agreement between the feature vectors during the testing phase.

$$\Omega(v) = \arg\max_{\omega} \sum_{\varepsilon=1}^{E} w_\varepsilon \, P_\varepsilon(\omega|v) \tag{3.20}$$

3.3.7 Cross-validation performance measures

Since there is a variable number of observations per class, the prior probabilities are not equal between classes. In other words, the dataset is unbalanced. As such, the accuracy is not entirely a reliable candidate as a metric for the performance assessment of a classifier. The confusion matrix [Provost et al., 1998] is a table which reports the number of false positives (FP), false negatives (FN), true positives (TP) and true negatives (TN) and allows a more detailed analysis. FPs are type I errors while FNs are type II errors. Table 3.3 shows the confusion matrix for binary classification.

Developed by electrical and radar engineers during World War II for detecting enemy objects, the Receiver Operating Characteristic (ROC) has been extensively used in the biometrics literature to select potential optimal models. The True Positive Rate (TPR) = (Recall) — total positives — and False Positive Rate (FPR) = 1 - Specificity = $FP/(FP+TN)$ — total negatives — are plotted against each other to visualize the performance of a binary classifier as the discrimination threshold is varied. Since ROC curves are only usable for AVA (face verification) scenarios, they can not be used in OVA multi-class problems to compare performance.

		Prediction		
		True	False	
Actual	True	TP (hits)	FN (misses)	Recall
	False	FP (false alarms)	TN (correct rejections)	Specificity
		Precision	NPR	Accuracy

Table 3.3: Binary classification confusion matrix

In the context of information retrieval, precision and recall are defined [van Rijsbergen, 1979] in order to describe the relevance of a result. Precision or Positive Predictive Rate (PPR) is computed as $TP/(TP+FP)$ and is an indication of the fraction of retrieved instances that are relevant to the query — the number of predicted positive results. In other words it is a fraction of correct results from the total number of results. Recall or Sensitivity, computed as $TP/(TP+FN)$ represents the probability that relevant instances are retrieved by the query. Precision and recall do not depend on TN, but only on the correct labeling of positive examples TP and the incorrect labeling of examples (FP and FN). These measures can provide a good perspective on an OVA classifier performance.

The relationship between (Precision-Recall) PR curves and ROC is explained in [Davis and Goadrich, 2006] concluding that an algorithm that optimizes the area under the ROC curve is not guaranteed to optimize the area under the PR curve. By varying the ratio between the first column and the second

column in table 3.3 one can plot Precision vs 1 - Specificity — achieving the same results as with ROC curves — or plot precision vs recall. For the current experiment, since we are only interested in relative performance, neither ROC or PR curves are reported.

Towards an indication of *effectiveness*, the $F - measure$ or $F - score$ (eq. 3.21), as commonly referred to in the literature, is the harmonic mean between precision and recall. Typically β is taken to be 1 hence the underscore notation F_1 score. This measure can thus provide an appropriate singular value which we can use to compare relative performance.

$$F(\beta) = \frac{(1+\beta^2)\, PR}{\beta^2 P + R} \tag{3.21}$$

As a generalization towards multi-class problems, precision and recall are computed for each class ω independently using the confusion matrix $\mathbf{M}$. To compute recall R_i we subtract the element on the diagonal (TP_i) for class ω_i from the horizontal sum of elements on that line as shown in table 3.3 to obtain FN_i. The $F_1(\omega_i)$ score is then computed for each class using eq. 3.21.

$$P_i = \frac{TP_i}{\sum_j^N \mathbf{M}_{j,\,i}} \tag{3.22}$$

$$R_i = \frac{TP_i}{\sum_j^N \mathbf{M}_{i,\,j}} \tag{3.23}$$

The performance of each dataset was evaluated using leave-one-out cross-validation (LOOCV), a special case of k-fold cross-validation, where $K = N$. In LOOCV, a total number of N training/testing folds, equal to the number of observations N are performed. For each fold, one camera observation is selected as a probe and the rest of the $N-1$ camera observations are used for training. As previously mentioned, each camera observation is divided in patches, hence each patch was marked as belonging to one unique frame, in order to know which observations to select for testing and training (patches are treated as sets). For each fold, the F_1 score is computed from the confusion matrix.

Once we have the F_1 score, precision and recall for all classes ω over all folds, then the quality of the overall classification for one dataset can be assessed in two ways: using either micro-averaging (eq.3.24) or macro averaging (eq.3.25). While micro-averaging gives equal importance to each camera observation, macro-averaging gives equal importance to every class. This is because the F_1 measure ignores TN and thus its magnitude is mostly determined by the TP [Forman and Scholz, 2010].

$$F_{1\mu} = \frac{2\, \sum_k^K P_\omega^k\, R_\omega^k}{\sum_k^K P_\omega^k + R_\omega^k} \tag{3.24}$$

$$F_{1M} = \frac{1}{K} \sum_k^K \frac{2\, P_\omega^k\, R_\omega^k}{P_\omega^k + R_\omega^k} \tag{3.25}$$

Since the number of observations per class is unequal we do not intend to give equal weight to each observation since we are performing LOOCV, thus the Macro averaging measure F_{1M}-score was used.

3.3.8 Experiment search space

Once having a singular value to measure effectiveness F_{1M}, the goal was to observe how parameters would influence the recognition rate for each feature type, grid size, number of clusters and number of neighbors used in classification, since these could be specific for each dataset. For each grid size, the number of clusters E was varied $\pm\hat{E}$ using the average number of patches per frame, as computed in the last line of table 3.2. $E_{5cm} \in \{6, 9, 12\}$, $E_{4cm} \in \{9, 12, 14\}$, $E_{3cm} \in \{16, 20, 25\}$ and $E_{2cm} \in \{36, 42, 49\}$. Since the number of clusters was varied, the number of neighbors used for classification was varied as well, by taking increasing odd values $\mathcal{K}_{nn} \in \{1, 3, 5, 7\}$ in order to avoid ties.

The F_{1M} score was recorded for comparison reasons for both one single KNN classifier and for the experts which were trained using the clustering from either SOM or K-means. Of course, in the case of one single

KNN the cluster labels did not make any difference since they were not taken into account. Although, initially majority voting was tested, after a cross-validation test on the SHOT 5cm dataset, the sum rule resulted in a 0.3% performance increase and as such was subsequently used.

3.3.9 Evaluation procedure

For each dataset the following procedure was repeated, a data flow schematic is given in figure 3.9 :

1) Clustering: The dataset is selected, shuffled and clustering is performed on the whole set minus the probe camera observation set. Face region labels are assigned to each patch. Then, each observation has three types of labels: the unique frame ID (camera observation), the class label and the newly obtained face region label. In the testing phase, the probe frame is selected and the distance to the nearest centroid is computed and each observation is assigned to a cluster. We could have forced to have unique cluster assignments for each face region by re-computing the distance to the duplicate face regions, however this was not the case and some experts were 'triggered' multiple times while others did not. In this way no data was discarded.

1a) The number of clusters E is selected for the specific E_{grid} and both SOM and K-means labels are obtained. Thus, in total, for each set, the clustering was performed $|M_{grid}| = 3 \times 2$ methods = 6 times. Hence, the two clustering algorithms are compared using the same number of clusters.

2) LOOCV: Having each observation labeled according to a face region and also knowing the class, E KNN classifiers are constructed independently per cluster. Each camera observation is selected stochastically only once from the whole set and used as the probe (testing). The rest of the data is used for building the classifiers and the process is repeated $N = 152$ times, until all the camera observations are used as probes. For comparison, one single KNN classifier is constructed simultaneously, using the same parameters, not taking into account the face region labels. Over all runs the F_{1M} score is computed as the final performance measure for both the experts and the single classifier. Thus, for each of the six sets in step **1a)** three measurements are taken: $F_{\mathcal{K}_{nn}} \in \{F_{1M}SOM, F_{1M}Kmeans, F_{1M}OneKNN\}$.

2a) The dataset is unchanged, however the number of neighbors is increased and step **2)** is repeated until all the values in $\mathcal{K}_{nn}$ are used. Then, the process goes back to step **1a)** — the number of clusters is increased. Once all the cluster sizes have been exhausted, everything is repeated from step **1)** with the next dataset, using a different feature descriptor and grid size.

3.4 Results

The overall experimental results are shown in fig. 3.10 which gives clear insight towards the relative performance of the descriptors according to grid size (X axis) and of the experts compared with one single classifier. In figures 3.11 and 3.12 all the F_{1M} scores are given, each plot showing the results for different sizes of the grid and number of clusters E. The columns correspond to the number of clusters E used while each row corresponds to a different grid size. In both figures the colors represent a different feature type while the clustering method is depicted with different line styles.

3.4.1 Features

The results of the experiments are plotted in fig. 3.10 where the Macro F_1 score is averaged over the runs with a different number of neighbors for KNN and number of clusters. It is clear that SHOT (red) is

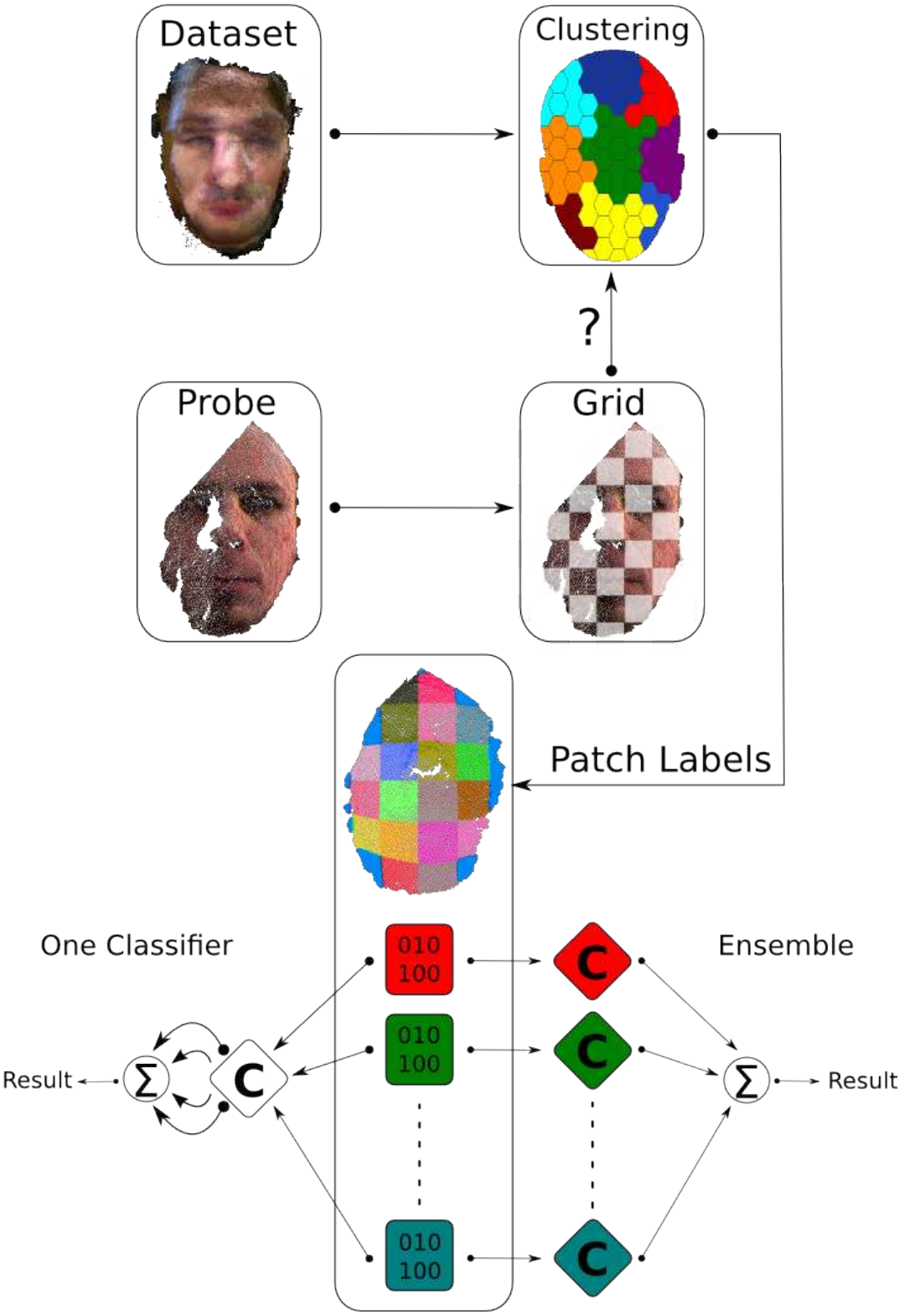


Figure 3.9: Illustration of the steps involved in classification. Clustering is initially performed on the whole dataset (patches). When a probe arrives, the clustering algorithms (SOM or K-means) assign each patch from the probe a face region label. Features are extracted and the results of either one classifier (Left, white C rhomboid) or several face region experts (Right, colored C) are combined using the sum rule (eq. 3.20) to obtain the final class label. Each color represents a unique face region cluster.

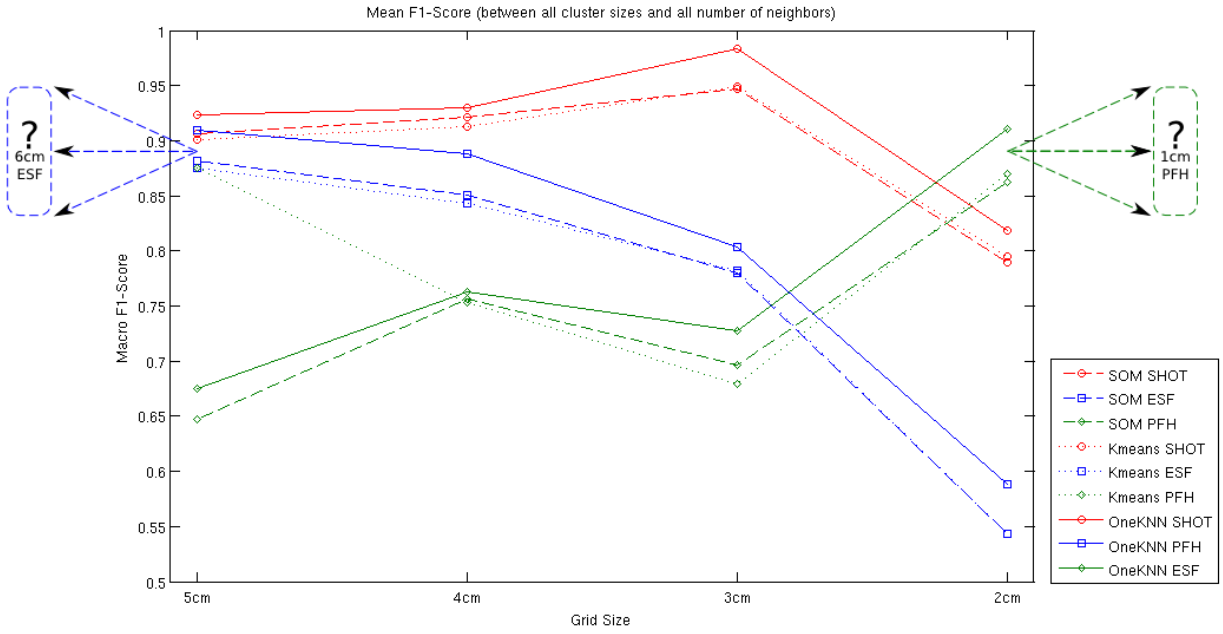


Figure 3.10: The average Macro F_1 score over all numbers of neighbors $\mathcal{K}_{nn}$ and the number of clusters used E. SHOT (red) is clearly the best overall, peaking at 3cm grid size. An interesting observation is the evolution of PFH (green) which increases in score at 2cm after performing the worst at larger scales. The question arises if the trend will continue at 1cm grid size. The opposite effect can be observed for ESF (blue) which might be more useful at a higher scale.

the best descriptor over all grid sizes, peaking at a grid size of 3cm. SHOT is a combination between signatures and orientation histograms, and as such it was observed to have the highest discriminative power between all three feature types.

The figure also suggests that PFH (green) increases its performance along with resolution. This is in accordance with the way the feature descriptor is constructed. Since its metrics are based on the neighbors within a certain radius, the closer the neighbors the better the description. This makes PFH a good candidate for a bag-of-features approach, since training an expert for each cluster might become inefficient for a large number of clusters.

The ESF (blue) descriptor also has a trend to increase in performance as the resolution decreases. This suggests that ESF might be a good weak rejector. However, it should be noted that during this experiment we might have just exploited the "specific" randomness within the feature descriptor, since the number of classes in the dataset is small. Furthermore, the descriptor was not tested against itself, for example after running the feature extraction algorithm multiple times on the same patches. Due to the nature of the descriptor it is highly unlikely to generate the exact same feature vectors on the same data, over multiple extraction steps. Taking a closer look at the SHOT and PFH histograms suggests that the feature vectors are sparse. This is most likely the case since the patches have a "square" shape and the feature descriptors perform radius based computations, hence some values will be zeroed out. This most likely causes the clustering algorithms to be stuck in local optima. Furthermore, the same reason might cause the drop in performance when the number of neighbors K increases. These issues are discussed in the next sections.

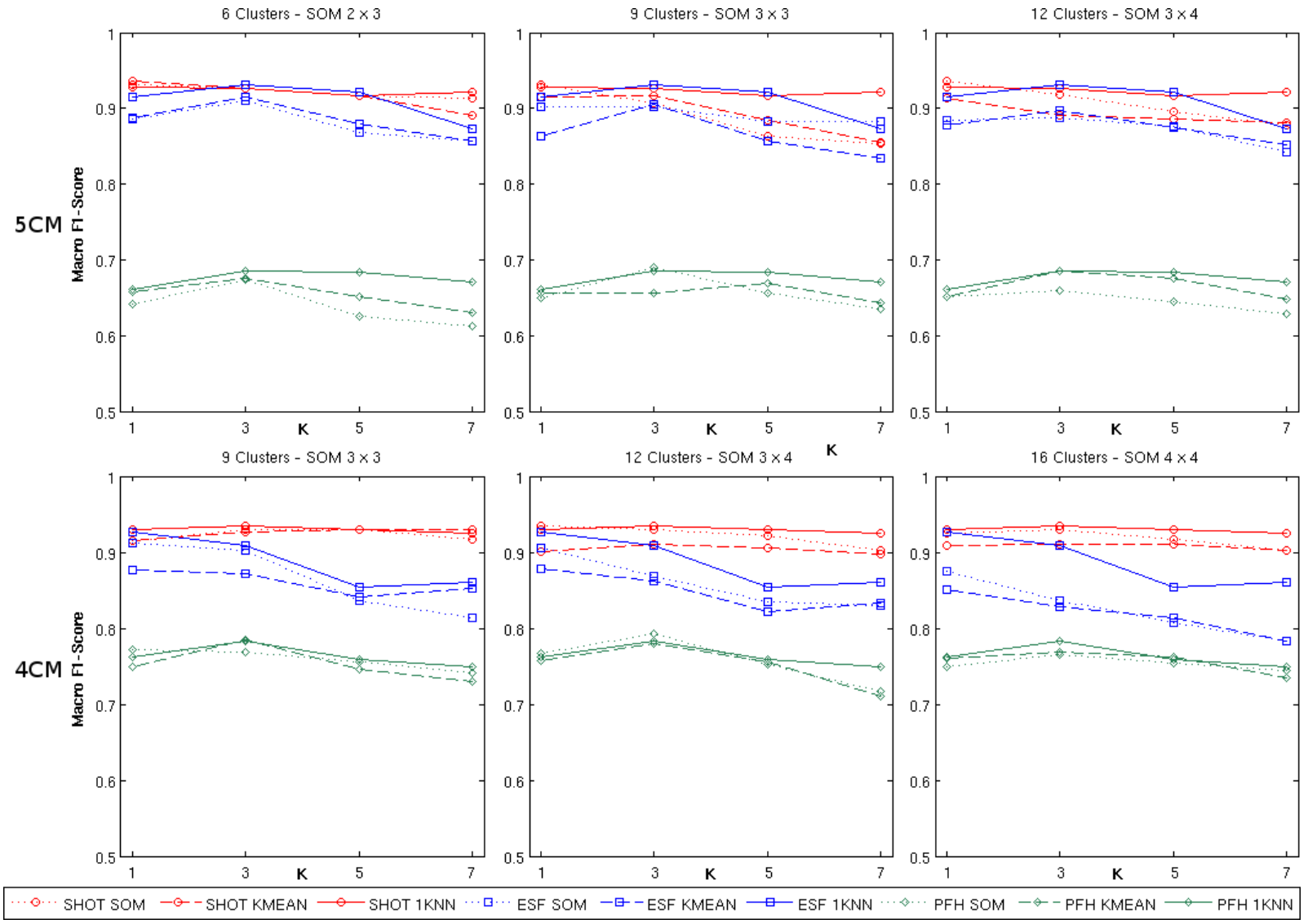


Figure 3.11: 5 and 4 cm grid sizes (rows) per cluster size (columns).

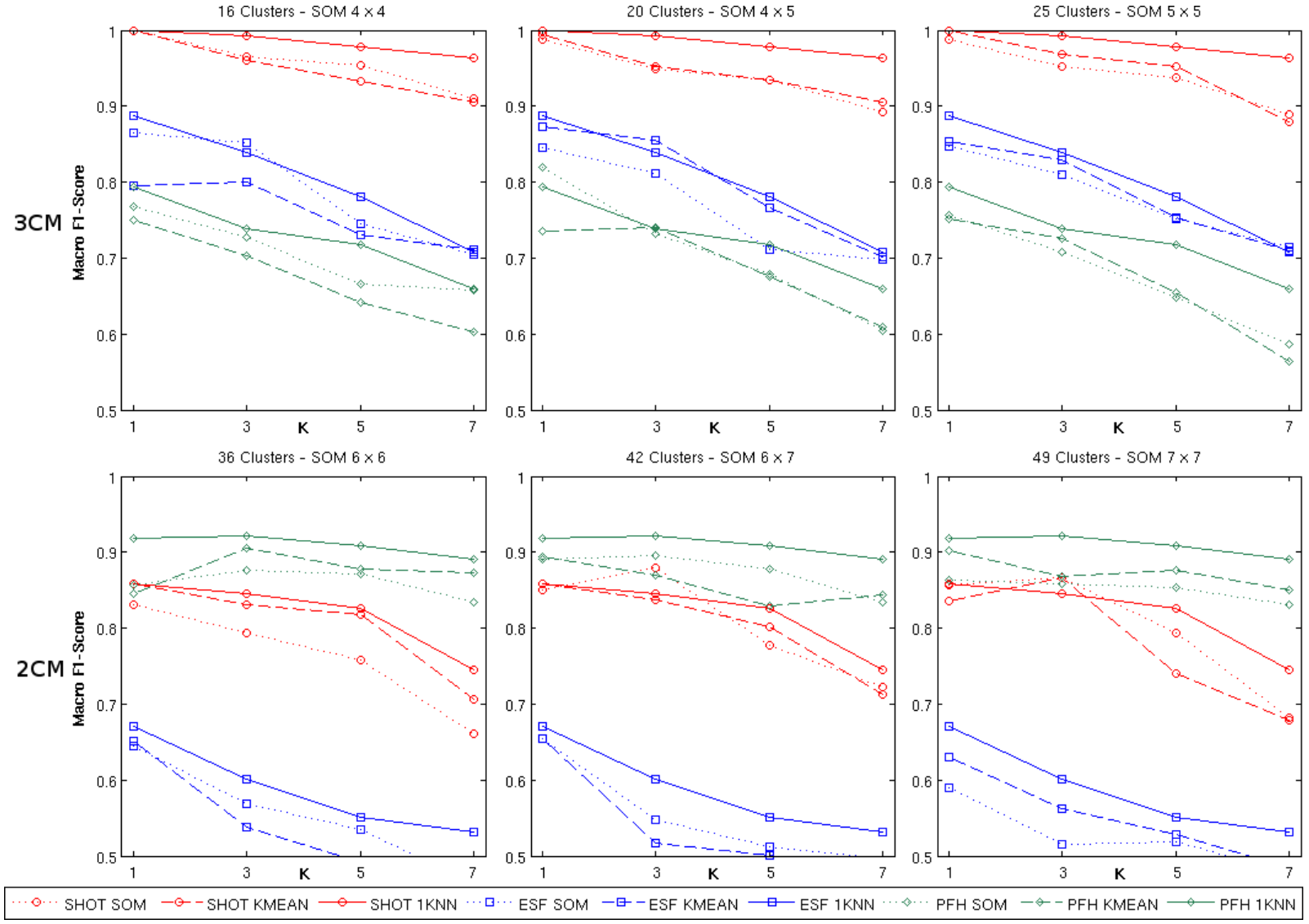


Figure 3.12: 3 and 2 cm grid sizes (rows) per cluster size (columns). On the top left figure, 100% efficiency was achieved, since this is a small pose normalized dataset. The decision boundary which does not generate any crossvalidation error was found for K=1.

3.4.2 Clustering

From fig. 3.10 we can conclude that the single KNN classifier (solid lines) — which does not take into account the face region labels generated by clustering — has a lower overall classification error than training separate experts for each face region. First of all, the dataset contains a small number of classes and thus one single classifier is more likely to outperform the experts, since it is trained with all the examples and the decision boundary is not too complicated. Moreover, the dataset is pose normalized which considerably decreases the complexity for a single classifier.

However, the main intuition why this would be the case is that the labels might not accurately separate the face regions. This would imply that the sample diversity between experts is too low, hence the advantage of combining classifiers is lost. Furthermore, as the grid size decreases, as we can observe from figures 3.11 and 3.12, the relative F_1 score between clustering and no clustering increases. A smaller grid size generates more observations, which implies that as the number of observations increases the convergence to the global optimum is more difficult since the search space is larger.

Following this hypothesis, the clustering in one case of the 12 datasets was investigated. We can observe from the plots in 3.11 and 3.12 that the best case for this premise would be when the F_1 score distance between the single classifier and the experts would be highest. The results from 5cm and 4cm do not have such high disparity for either SHOT or PFH. Since SHOT was the best performing descriptor, the case of 3 cm with $E = 20$ clusters was chosen — fig. 3.12 first row, second column. In the case where the number of neighbors $K = 3$, both SOM and K-means clustering methods partition the dataset sub-optimally (e.g. the subsets are not disjoint enough). The selected dataset contains 3254 bservations with an average of 21 per frame (table 3.2).

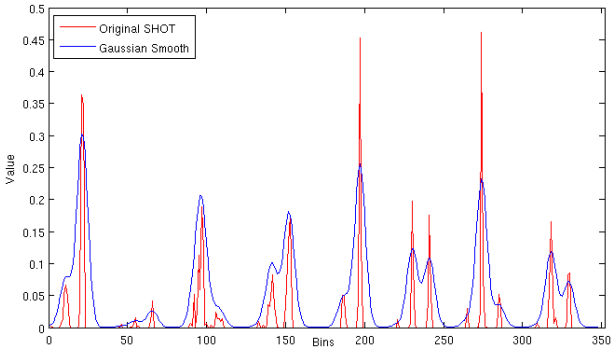


Figure 3.13: Gaussian smoothing on a 3cm SHOT histogram. $\sigma = 0.01\%$, window size = 0.1%

In order to evaluate the clustering qualitatively, PCA was applied to the dataset and the most significant three principal components were kept. PCA reveals the internal structure of the data in a way that best explains the variance in the data. The data points are rotated around their mean in order to align them with the principal components. Due to the orthogonal transformation, most of the variance is moved into the first three dimensions. Usually, the values in the remaining dimensions do not describe much of the variance and can be discarded with minimal loss of information. We can not guarantee that this is the case here as well, however, it is a good strategy to use for visualization purposes.

The projected data was plotted in a 3D scatter plot (fig. 3.14) where each of the 20 clusters is depicted by a different color. A Gaussian smoothing operation, equivalent to a low pass filter was applied to the dataset. The window size was set to 0.1% of the feature size and $\sigma = 0.01\%$. In figure 3.13 one typical 3cm SHOT histogram is shown before (red) and after (blue) the moving average operation was applied.

The figures in the top row (fig. 3.14) are the clustering result before the smoothing operation and the ones in the bottom row after the smoothing process. In the first row, the plot seems to form a tetrahedron, with most of the clusters concentrated at the corners and a few sparse clusters in the middle. In the lower plots, the data points are distributed more evenly and there is also less overlap between the clusters. After LOOCV on the smoothed dataset, the F_{1M} score of the experts was equal to the single classifier even when the number of neighbors was increased.

All of the 152 original frames have a very similar spatial overlap due to the pose normalization. The

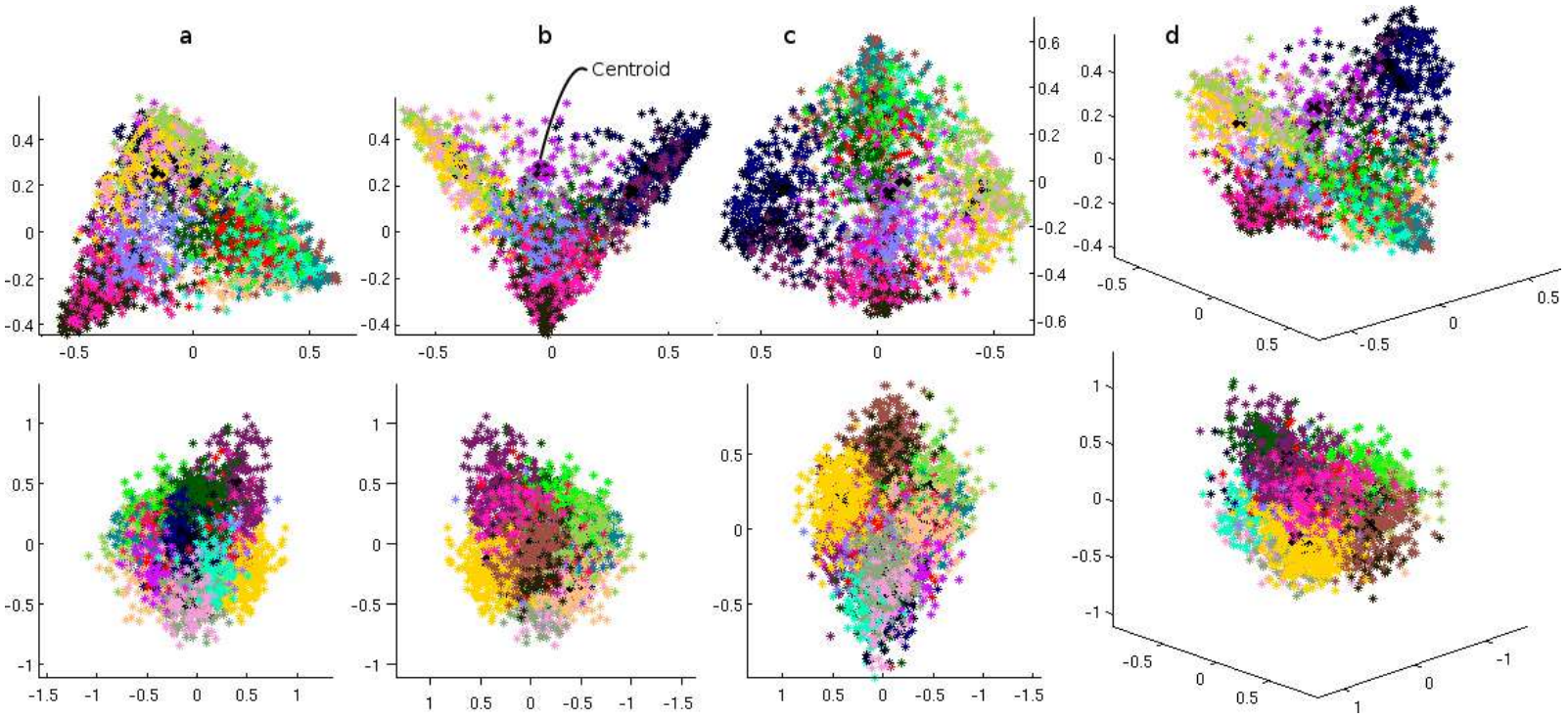


Figure 3.14: Best viewed in color. Visualization of the clustering result (K-means) of the 3cm SHOT dataset using PCA projection to three dimensions. In the first row the data forms the shape of a tetrahedron. Most of the clusters share the same corners with only a few clusters in the middle. In the second row the moving average operation was applied (fig. 3.13). The data is spread out more evenly in form of a sphere. The clusters are better separated and can be visually distinguished much easier.

amount of missing data is not large — the dataset contains mostly frontal views. This would imply a distribution of the number of patches per cluster with a small deviation from the mean. Furthermore, the question arises whether all experts are trained with examples from all the classes. By taking a look at figure 3.15 (numbers above bars) we can observe that some clusters do not contain examples from all the 8 classes, thus the posterior probability of some classifiers on evaluating some classes would be zero.

For a lower number of clusters, SOMs produced a more evenly distributed number of unique class names per cluster, while K-means had higher variance per cluster. When the number of clusters was increased, K-means tended to converge more often to a better solution, although the final solution was stochastic. However, after the smoothing process, the number of clustering solutions decreased — the search space was reduced. Two examples are given in figure 3.15, the profile of the blue line could be considered as a "signature" of the clustering solution. Another way of reading fig. 3.15 is to make a parallel with the bag of features approaches: here all the histograms per camera observation are averaged into the blue line. In other words, these figures can be considered as the overall dataset clustering codebook.

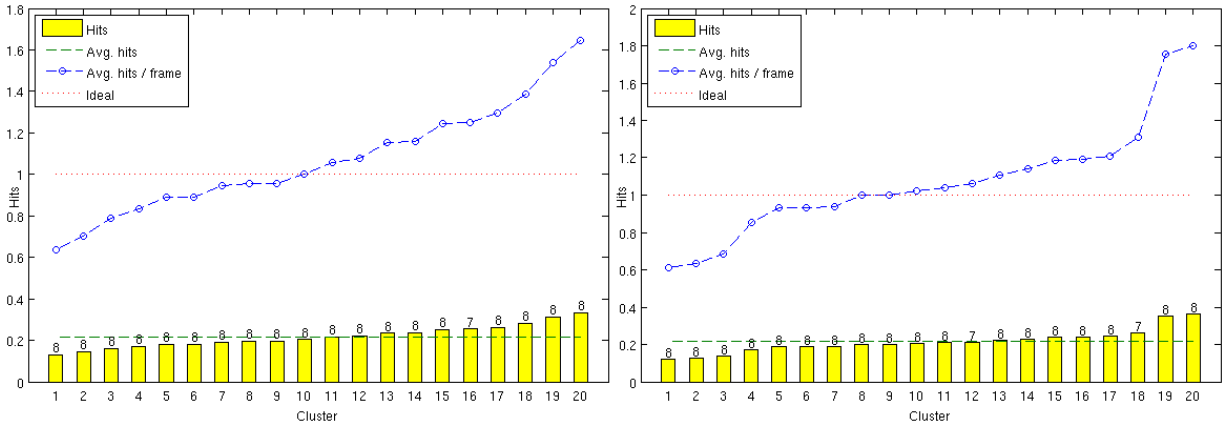


Figure 3.15: Two outcomes to clustering (K-means). The yellow bars are an indication of how much data is used to train each expert (sampling from the whole dataset according to the clustering result) and should be read as relative to the other bars and not the vertical axis. The green dotted line is the average between all bins (N / E). The values were sorted in ascending order to facilitate comparison between the two outcomes, hence cluster names do not correspond between figures. The unique number of classes from each cluster subset is given above the yellow bars. We can observe that the experts are not trained with an equal number of examples (e.g. green line). Furthermore, the number of unique classes that an expert will learn (shown above the bars) is not always the maximum (left: expert 16, right: experts 12 and 18). The blue dotted line can be considered the "clustering signature" of a clustering outcome. In order to plot this line, the bag of words (word frequency) for each camera frame was computed. Then the frequencies were summed for all camera frames. Since we are sampling uniformly, the red line would be the ideal case where each camera observation would have a unique number of hits per camera frame — e.g. almost equal frequency for each word, between frames.

In essence, what figure 3.15 tells us is that the closer the blue line is to the red dotted line, that is, the lower the slope, the better the clustering solution. Ideally, each complete face observation would have unique face region labels. That is, there could only be one left eye, one mouth, one nose, etc. (depending on the grid resolution). As previously mentioned, having unique face regions for each camera observation could have been forced by computing the distance to the second closest centroid for already assigned face regions and repeating the process for all duplicates (this algorithm is tested in the next chapter). This was not the case in this experiment since the number of clusters was varied and for a small number of

clusters E it would not have been possible to generate unique distributions for each camera observation.

When looking at the blue line in fig. 3.15, we can see that despite the uniform sampling method (e.g. grid to patches), we do not have a uniform distribution of the training data after clustering. This is also partly due to missing data. After clustering, some frames have more labels of a particular kind. From the figure on the right (fig. 3.15) one can observe that most of the central clusters are close to 1. This is an indication that the experts trained on those face regions are "activated" during testing about one time on average for each frame, which suggests that the cluster is indeed representative of a specific face region. As such, one can apply weights to the sum rule in the case of experts to give more importance to the more accurately identified clusters or experts.

The weighting could also be defined as proportional to the probability of an expert to make a right decision, computed over the mean squared error (MSE) between all classification cases. This information could also be used as an objective measure to supervise the clustering algorithm or to select an optimal clustering solution. Moreover, it could be combined with topological heuristics, to ensure that each face region is assigned only if it originates from the right face region. This process would require labeled face regions (true position).

3.4.3 Classification

The main difference between the effectiveness of one single classifier versus the experts is directly influenced by the clustering. As discussed in the previous section, in the case of SHOT, by smoothing the histograms the clustering was improved which in turn resulted in a higher diversity between clusters. The F_{1M} score was computed over three LOOCV runs using different number of neighbors $\mathcal{K}_{nn}$ on the SHOT 3cm dataset after smoothing. The performance was high even when the number of neighbors was increased (table 3.4), which indicates good generalization capability. With larger grid sizes (5 and 4 cm) the number of neighbors do not influence the result much. However, when the resolution increases, more neighbors imply a lower F_1 score, especially for ensembles architecture, because the clustering solution converges towards better local optima or perhaps global optima.

$\downarrow F_{1M}$ | K $\rightarrow$	1	3	5	7	9	11
SHOT 3cm	1.0	0.9776	0.9857	0.9711	0.9510	0.9280
PFH 2cm	0.9371	0.9464	0.9628	0.9560	0.9503	0.9503

Table 3.4: First row, a Gaussian filter is applied to the SHOT 3cm dataset. Second row, PFH is normalized using the z-score $(x - \mu)/\sigma$. Both methods result in higher performance. Smoothing SHOT features minimizes the euclidean distance between feature vectors. Standardizing PFH feature vectors suggests that the correlation distance might be a better metric.

3.4.4 Conclusions

During this small scale experiment we have proven that it is possible to achieve high classification accuracy using a decompositional approach towards face recognition. Despite the weak object recognition feature descriptors which were used to represent data, this method is robust towards partial occlusions and missing data. Due to the non-holistic description of the face this method shows flexibility towards other applications (e.g. identification of expression, age, gender, accessories, etc).

Furthermore, we have shown that it is possible to identify face regions even without supervision. Subsequently, in order to simplify the decision boundary, several experts have been trained for each face

region. Given the low computational complexity of the architecture and considering pose normalization as approximate, this method can be applied to real-time tasks. A possible application could be small-scale biometric authentication, since during the enrollment and evaluation, subjects have near-frontal poses relative to the sensor.

For this coarse normalized 8 class dataset, we have identified the optimal grid scale and the most promising feature descriptor for this application. Two unsupervised learning algorithms were evaluated for the identification of face regions and showed similar results. The SHOT feature descriptor performs spherical computations while all the patches are square-shaped surfaces, which generates zero-valued bins. This is a consequence of the uniform patch-based sampling method. Applying a Gaussian filter over the feature vectors resulted in better separation of clusters and, in turn increased expert classification effectiveness. The standardization of data suggests that correlation distance is a better metric.

The simpler method of using one single KNN classifier and combining results using the sum rule outperformed the face region experts in almost all cases. This was the case since the clustering was sub-optimal and there was not enough diversity between experts. The possibility of combining feature experts arises since the PFH descriptor suggests better performance at higher sampling resolutions. Also, ESF could be used as an initial weak rejector. Moreover, weights could be added to the sum rule (eq. 3.20), learned from the clustering "signature" (fig. 3.15), the overall MSE for each expert or by training a neural network as meta-classifier, similar to the mixture of experts [Jacobs et al., 1991] algorithm.

The current dataset is too small to get an insight towards the generalization capabilities with a higher number of classes and pose and expression variance. Hence, in the next chapter we keep the best feature descriptor and architecture and test a different sampling method on the complete dataset which contains more classes with pose and expression variance.

Chapter 4

Viewpoint invariance without normalization

From previous research [Queirolo et al., 2010, Kakadiaris et al., 2007, Faltemier et al., 2008a], we can conclude that pose normalization is computationally complex and is the main computational bottleneck. Thus, we propose a novel computationally economic sampling method which does not require pose normalization of each observation in order to extract data. In this way, the features are computed in similar regions where the surface curvature is most invariant to rotations or translations, regardless of pose, using the Scale Invariant Feature Transform (SIFT) [Lowe, 2004] keypoint detection algorithm.

The previous dataset consisted of pose normalized camera observations with minimal alignment error. As previously mentioned, the pose normalization over the whole dataset (table 2.1) is not consistent, with Freestyle not being pose normalized at all, containing extreme pose angles. Furthermore, each type of dataset (yaw, pitch, roll, etc.) contains missing data which is specific to the pose. As such, the uniform sampling approach in the previous chapter can not be successfully applied to the whole dataset and a different means of extracting data is necessary. Since each frame contains 13K data points on average, we require a way of identifying the regions with the most significant information.

Since zones with low local curvature — depending on the sub-sampling size — do not bring much information towards the discriminability of faces, these areas can be discarded, thus saving computation time. However, the global topological relationship of all the face regions is still important. As such, the location of each extracted region is also recorded.

4.1 SIFT keypoints

The SIFT algorithm detects keypoints in regions with high-contrast. In the 3D version of the algorithm the contrast is equivalent to the surface curvature. The algorithm detects keypoints with consistent positions over the same objects in order to deal with changes in image scale, noise, translations or rotations and in our case, pose.

Since faces have the same overall intrinsic shape, we propose the SIFT keypoint extraction method for data sampling. This allows us to extract feature vectors which lie in similar regions of the face, regardless of pose, thus bypassing the normalization step. To our knowledge this has not yet been attempted in the 3D face recognition literature. An additional motivation behind using SIFT as a sampling method is the

fact that the dataset contains observations with partial occlusions and / or missing data. The inherent nature of keypoint extraction in SIFT ensures that similar regions are sampled, regardless of occlusions or pose. This novel approach thus has the potential of solving the difficult problems of noisy data, pose variance and occlusions.

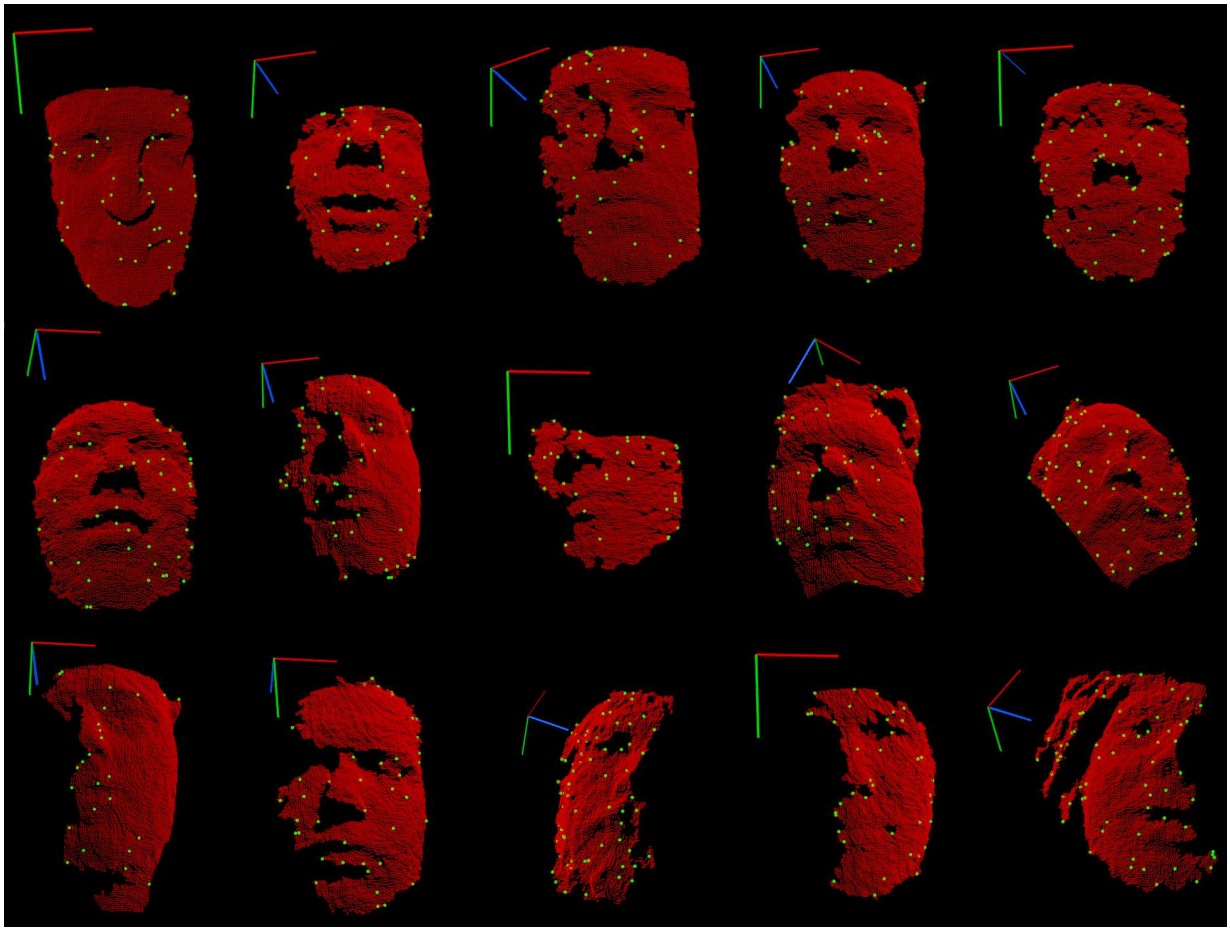


Figure 4.1: Camera observations from several subjects. The SIFT keypoints (green) are detected in similar face regions. The camera observations come with missing data and are also pose variant, which can be observed from the reference axis: red = x, green = y, blue = z. The advantage of using SIFT is that features are extracted only in areas with high curvature thus eliminating the need to compute features for the whole set (full camera frame).

A qualitative assessment reveals that the keypoints have the tendency to form around edges since there is a high degree of curvature due to sensor noise. This is not to our advantage, however these keypoints are likely to cluster together (see section 3.2, p. 23) and thus can be assigned less importance if the corresponding expert has weak discriminative power.

The distribution of the keypoints depends on the parameters used for SIFT: according to the minimum scale, the number of octaves and the number of scales per octave, the result can have high precision and low variance or vice-versa, which affects the consistency of the data sampling location.

When the precision is high, the keypoints are detected more accurately around the same regions between

frames — for example a keypoint is either detected or not in the middle of the eyebrow. This was the case when the minimum scale was set to 0.3 cm with 5 octaves and 10 scales per octave, which resulted in sampling an average of 15 keypoints per frame, mostly around the eyebrows and the nose, with some regions completely lacking keypoints. When the precision decreases, there is less consistency of the location of keypoints between frames, however this results in an average of 50 keypoints per frame — for example, three keypoints will be detected around the eyebrow. For the images in fig. 4.1 the minimum scale was set 0.4 cm with 4 octaves and 5 scales per octave. In both cases the minimum curvature was set to 0.1 cm since this provides a large initial keypoint sample for SIFT, however it is also a source of error since it allows keypoints to be selected from noisy fractal-like regions as well. No attempts were made at preprocessing the original camera data in order to interpolate or smooth the surface.

Essentially, the tradeoff has to be made between high sampling consistency with less keypoints or medium consistency with more keypoints. In the case of a live camera stream the first case might be a better option since the data is abundant. We have opted for less consistency, even if introducing more noise, in order to have more samples.

Since the camera observations are pose variant, the parameter selection can not be done quantitatively unless the pose is normalized or the variance is analyzed over categories of pose variance. The SIFT sampling method causes keypoints to cluster in particular regions (fig. 4.1). As such, the feature vectors from a particular region/cluster are also similar since they are extracted from neighboring locations. In this sense, this sampling method could be compared with a sliding window approach since we are extracting multiple feature vectors from the same region, which in turn increases the probability of finding a match for the probe in the dataset. Hence, a "less consistent" distribution is similar to parzen window based approaches.

While pose normalization is computationally expensive for the whole frame, it can be efficiently done for the keypoints. As such, once the keypoints are detected they were aligned to a 3D face template (fig. 2.6) using the ICP algorithm. The new coordinates are recorded along with the extracted feature vectors. While this approach does not guarantee that the alignment is perfect, it allows us to take advantage of the approximate topological sampling location. It is evident that the normalization of the keypoints can not be entirely accurate since camera observations with extreme viewpoints or a large amount of missing data do not have enough correspondences between the keypoints and the template.

Figure 4.2 shows the keypoints obtained from a randomly selected subset of camera observations. For visualizing, the keypoints are computed for all frames, then the keypoints are simply drawn together in the same window. On the right the keypoints are pose-normalized using ICP, showing a higher consistency than the left frame that shows the original topological positions.

4.2 Feature descriptors and sampling methods

From the previous chapter we have observed that the SHOT descriptor had the highest discriminative power at a patch size of 3cm. The radius was kept fixed for all descriptors at 5cm while the size of the patches was varied. However, since we are no longer using patches (e.g. cropping) to extract features, the radius of the feature descriptor was re-evaluated. The previous 8 class dataset was used for this for comparison with the previous sampling method.

The efficiency of the descriptors was evaluated using LOOCV on the initial 8 class dataset (page 29) using features extracted using PFH — where the keypoints were uniformly sampled and for ESF where the complete frame was used. For SHOT the sampling was made using SIFT. The results are shown in table 4.1. Since the PFH descriptor is based on the relationship between all pairs of the points in the

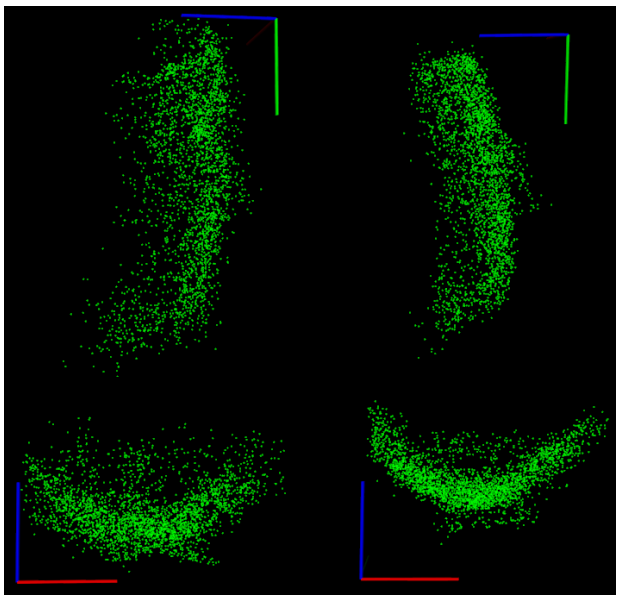


Figure 4.2: SIFT keypoints are computed for a randomly selected subset of camera frames. The left side shows the original keypoints. On the right the keypoints are normalized using ICP with a common reference frame of a sub-sampled face template (fig. 2.6). The first line is side view, bottom line is a top view. On the right the keypoints show less variance in position and suggest more consistency. The positions of the keypoints on the right were recorded prior to the feature extraction, as the topological origin of the feature vector.

neighborhood, the computational complexity is $\mathcal{O}(n^2)$. Thus in order to achieve similar performance as SHOT, the PFH descriptor requires considerable more processing time.

The results of the experiments in the previous chapter implied that the SHOT feature descriptor performance was optimum at a 3cm patch size and suggested that PFH might perform better at 1cm. Conversely, while observing the results in table 4.1 we can see the opposite effect: performance increases with radius size. While this might be counter intuitive at first, let us consider the differences between the previous and current method of sampling.

In the previous experiments the data was uniformly sampled and cropped (rectangular regions — patches) and we observed best performance at 3cm patches with the feature descriptors radius set at 5cm. The same sampling resolution for the uniform grid was used in this experiment for PFH (e.g. 3cm grid). In the case of the SHOT feature descriptor which separates the spherical sampling region in bins, it resulted in vectors with empty values (e.g. imagine a square inscribed in a circle) and as such smoothing resulted in higher accuracy.

The increased discriminative power might have come from the fact that the most descriptive power is towards the center of the sphere in SHOT descriptors and using cropped data. Furthermore, on such a small dataset, the cropping might cause random measurement errors, which can generate distinct patterns in the feature vectors, which can then in turn increase discriminability.

Since the dataset consisted of pose normalized faces and the SHOT features were uniformly sampled over a grid of patches (increased likelihood of overlap between camera observations) it was the case that the optimal performance was observed at a patch size of 3cm. In the current case, where SHOT features are computed using SIFT keypoints, the performance is higher when the radius increases since there is

SHOT r =	2cm	4cm	5cm	6cm
	0.5765	0.9766	0.9779	0.9857
PFH r =	0.5cm	1cm	2cm	3cm
	0.2011	0.3446	0.8728	0.9920
ESF	full camera frame: 0.8623			

Table 4.1: LOOCV performance on the initial pose normalized 8 class dataset using SIFT keypoints to extract SHOT features, PFH using uniform sampling with a grid of 3cm and ESF on the whole camera observation. The SIFT sampling method using SHOT feature descriptors showed the best balance between computational complexity and performance. For SHOT and PFH the sum rule was used to combine the results of one single KNN with K = 3. It takes considerably more time to compute PFH features with $r = 3$ cm than with $r = 2$ cm. While ESF is the fastest to process, SHOT with $r = 6$ cm is still faster than PFH with $r = 2$ cm. Due to the sampling method, PFH sees more data, hence the higher accuracy.

a higher level of overlap between the feature vectors when the radius increases. Furthermore, this is in accordance with the previous discussion of the consistency in SIFT keypoint detection. While it is favorable to use uniform sampling when there is no pose variance — higher overlap between frames, this would not be the case when pose variance is present.

Furthermore, when sampling uniformly, there is more data, however when using SIFT for keypoint extraction we are sampling only from the regions with most information, hence some information might be lost with the advantage of computational economy.

In the case of the PFH descriptor, which computes pairwise distances, a higher sampling resolution with a low region of interest resulted in higher accuracy. This is intuitive since with a large number of densely sampled features there is a higher likelihood of finding correspondences between camera observations. However, when the observations are pose variant, uniform sampling does no longer guarantee the same number of correspondences. Nevertheless, as the previous experiment suggests, the PFH descriptor could be used to extract features with a radius of 1-2cm on a densely sampled grid of 1cm or less. However, our intention is to focus on the larger shape context here.

Surprisingly, despite the fact that ESF performs computations based on randomly selected point groups, it achieves a performance of 86% on an 8 class frontal pose dataset. The effect of introducing more classes and pose variance might however cause a considerable drop in performance.

From table 4.1 we can conclude that the performance, peaking at a 6cm radius for SHOT, is similar to the results achieved in the patch experiment (previous chapter). Hence, in the next experiments the designated descriptor was SHOT with a radius of 6cm. Furthermore, the authors [Tombari et al., 2010] also suggest that the optimum radius is 6cm, determined empirically in the context of object recognition.

4.3 Clustering revisited

Since we are extracting feature vectors using a different method, we present two clustering experiments which have the potential to optimize the K-means clustering towards a better solution and in turn increase the diversity between the experts.

4.3.1 Keypoint location and feature vectors

As previously mentioned, the position of each keypoint was recorded for each feature vector. The most challenging of the recorded datasets was "Freestyle" which contains all types of variances, e.g. pose and expression, distance from sensor. As such, the performance of clustering using XYZ keypoint data or feature vectors for 36 clusters was compared. Furthermore, both the XYZ data and the feature vectors were length normalized and concatenated and used as input for the K-means clustering algorithm. The performance was evaluated during LOOCV using the sum rule of $E = 36$ KNN experts, with the results displayed in table 4.2. The results are lower (when compared to tables 4.4 and 4.5) since the euclidean distance is used here, there is a lower number of clusters which generates less disjoint training sets for the experts and all the 18 classes are used.

	XYZ	$\vec{V}$	$XYZ + \vec{V}$
F_{1m}	0.6909	0.7056	0.7547
F_{1M}	0.6401	0.6438	0.6995

Table 4.2: The Micro (F_{1m}) and Macro (F_{1M}) F-scores for LOOCV on the Freestyle dataset containing 18 classes with all types of variances. The concatenation of both keypoint extraction location and the feature vector (last column) outperforms either method. The Micro scores give equal weight to each observation and as such it is an indication of how many probes were correctly identified out of the whole set. The Macro score is lower since it gives equal weight to each class, in which case there is a greater penalty for a misclassification. The dataset consists of 61294 observations with the SHOT descriptor ($\vec{V}$) having 352 features.

4.3.2 Unique cluster hits per query frame

In the previous chapter we observed that when comparing the feature vectors with the K-means centroids, we did not obtain unique labels. Hence, we have decided to re-evaluate clustering on the original dataset while forcing unique cluster labels for the probe camera observations on the current dataset. However, here we no longer have a uniform sampling grid of features. Instead, the clustering is performed based on the SHOT feature vectors which are computed on SIFT keypoint locations. Since the parameters used for SIFT, as discussed in the previous section generate less consistent keypoints between camera observations, the requirement of unique cluster hits does not seem necessary. The greedy algorithm used for forcing unique clusters per observation is described below:

1) The probe frame is selected and for each feature vector the distance to each K-means centroid is evaluated and each feature vector is then assigned to a cluster.

2) If there are duplicates, the duplicates with the closest distance to a centroid and the unique hits are kept in a list.

3) For the remaining duplicates the distances to the clusters which have not already been assigned are recomputed and each duplicate is thus assigned to the available clusters with the closest distance.

4) The list is updated and the process is repeated from step 2 until there are no more duplicates.

However, a special case occurs when the probe contains a number of feature vectors larger than the number of centroids used during the clustering training procedure. Increasing the number of clusters (perhaps to the maximum number of keypoints registered for one frame for the whole dataset) would not be a good approach since some experts would be seldomly used since most camera observations are not outliers and have $E \pm 5$ keypoints. As such, two different strategies were tested.

In the first case "unique partial", for the frames which have more keypoints than the number of centroids, during step 4, the process is stopped if the list with the unassigned cluster centroids has been exhausted. This results in most frames having unique cluster labels and the ones with more keypoints having a partial unique cluster labels assignment, based on the closest distances to each centroid. For these camera observations, the remaining feature vectors are left as in the initial assignment and are therefore summed with the result of the corresponding expert for that particular frame. In this way no data is discarded. The second strategy "unique where possible" was to simply leave the original cluster assignment for the camera observations with a larger number of feature vectors than the number of centroids, while for the others the algorithm was applied.

The results of the LOOCV on the original 8 class roll dataset 3.2 for these two methods as well as the original method are displayed in table 4.3. The parameters were set to $E = 50$ clusters — KNN experts using $K = 3$ neighbors and a radius of 2cm — which is why the results are lower — using the SHOT descriptor. Using a smaller radius generated feature vectors which had less overlap, however also generated less accurate descriptions of the surface.

	Original (Closest)	Unique where possible	Unique partial
F_{1m}	0.8374	0.7960	0.6240
F_{1M}	0.7858	0.7484	0.5776

Table 4.3: The result of LOOCV using a greedy search for unique cluster centroid assignment. In the second column, the probes with more feature vectors than the number of centroids have partially unique solutions while in the third column they are simply left as original. The original clustering method, where each feature vector has a cluster label as the closest centroid (not forcing unique clusters) is given for comparison. For an explanation of the relevance of F_{1m} and F_{1M} scores, refer to section 3.3.7 on page 35.

As we can see from the table above, the original method of simply keeping the distances to the closest centroids still results in the best performance. This might be the case since we are using SIFT to extract keypoints and the distribution of keypoints is no longer consistent between frames. Most importantly, since the keypoints are usually in close proximity to each other it is not favorable or intuitive to enforce unique cluster labels since we usually have feature vectors coming from very similar face regions and as such, they should be assigned to the same centroid. Consequently, in the following experiments, the original clustering method was used and the duplicates were not discarded since they contribute as well to the final decision in the sum rule (eq. 3.20).

4.4 Parameters

The same method used in the previous chapter (see fig. 3.9) was applied using SHOT features computed at locations designated by SIFT keypoints. The main difference is the sampling method. In all cases the normalized $XYZ + \vec{V}$ was used for clustering (refer to table 4.2). Since the number of detected keypoints vary for each camera observation and features can be computed from keypoints which are in close proximity to each other (fig. 4.1), this does not imply that we should have unique feature subsets per frame, that is, to have unique cluster "hits" (refer to table 4.3). The parameters for the KNNs and K-means clustering were selected during a 10 fold cross-validation on the Freestyle dataset. As such, the number of neighbors was set to $K = 3$ to generate smoother decision boundaries which imply higher generalization on simpler sets hence less overfitting on the training set. The number of clusters was set to the average number of keypoints detected for this dataset, namely $E = 50$ in all cases. It should be noted that these are not necessarily the best parameters for the whole dataset.

4.5 Cross-validation results

In order to observe the difficulty per dataset, a 10 fold cross-validation was performed for all dataset using both methods, using $K = 3$ and $E = 50$ for the ensemble of experts. The results are displayed in tables 4.4 and 4.5 below. The average error is between $\pm 2 \times 10^{-2}$ with a 99% confidence.

Dataset	Yaw	Pitch	Roll	Z-Translation	Expressions	Freestyle	Average
Observations N	26722	29632	20707	41093	28774	61294	34703
Total N=208222	0.9306	0.9762	0.9853	0.9827	0.9633	0.9173	0.9592

Table 4.4: Single KNN and sum rule. The 10 fold cross-validation results on the individual sets. The most difficult, as expected is the Freestyle dataset, which is not pose-normalized at all. The next most difficult set is Yaw, where the translation of frames is slightly solved during normalization. The easiest set is Roll where the frames are almost frontal and the rotation variance is solved.

Dataset	Yaw	Pitch	Roll	Z-Translation	Expressions	Freestyle	Average
Observations N	26722	29632	20707	41093	28774	61294	34703
Total N=208222	0.9250	0.9707	0.9853	0.9787	0.9615	0.8995	0.9534

Table 4.5: Ensembles of KNN experts. The 10 fold cross-validation results on the individual sets. When compared to the single expert architecture, the accuracies are slightly lower, identical for Roll. We can notice a decrease of almost 2% accuracy on the Freestyle set, while the other results have decreased only by 0.005% on average.

4.6 Pair tests — a classifier's nightmare

In this series of tests, the previously described classification architectures (page 38) are trained on a dataset containing one type of variance and tested on all other datasets. Since the datasets are very different, we do not expect high performance, however we can observe the similarities. In total, there are 36 pair tests. The classes containing glasses are removed from the dataset in table 2.1 on page 19. Then, the SHOT feature vectors are computed for each camera observation. Each feature vector has a length of 352 features. The total number of feature vectors as extracted using the SIFT keypoint sampling method is given in the results table as the second row, below the subset name.

4.6.1 Sum rule and one classifier

The simplest method was evaluated first. In this case, there is only one expert C which is trained and subsequently classifies all the subset features of one camera observation $\mathcal{O}_{1..N}$. The posterior probability vectors for all the feature vectors are combined using the sum rule 3.20 and the class with the highest support is used as the final classification decision. The results for all the pair tests are displayed in table 4.6. The accuracy is reported with a 99% confidence interval, with the error in the order of $\pm 10^{-3}$ for all following pair test tables.

At a first glance over the results it seems that the accuracies between the pairs are not symmetric. The last row, containing the average testing error shows a similar pattern to the cross-validation results: the lowest average testing accuracy is on Freestyle, followed by Yaw, with the highest on Roll. The absolute highest accuracy is observed when the Translation dataset is used for training and Expressions and Roll are used

Testing → Training ↓ N =	Yaw 26722	Pitch 29632	Roll 20707	Z-Translation 41093	Expressions 28774	Freestyle 61294	Average Train
Yaw	-	0.4505	0.7628	0.5973	0.5211	0.4911	0.5646
Pitch	0.3490	-	0.7628	0.7107	0.7358	0.3201	0.5757
Roll	0.4859	0.6392	-	0.6707	0.6606	0.3395	0.5592
Z-Translation	0.3921	0.6593	0.8020	-	**0.8367**	0.3549	0.6090
Expressions	0.4090	0.6172	0.6259	0.6707	-	*0.3160*	0.5278
Freestyle	0.5685	0.4597	0.5183	0.5147	0.4495	-	0.5021
Avg. Test	0.4409	0.5652	0.6944	0.6328	0.6407	0.3643	0.5564

Table 4.6: Sum rule over one KNN classifier with $K = 3$ is trained using the euclidean distance. The accuracy is reported for all the pair tests.

for testing. The confusion matrix for Translation versus Roll is given in table 4.7 and for Translation vs Expressions the precision and recall per class is given in table 4.8.

Actual ↓	Prediction → 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	17	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
2	0	12	0	0	0	0	0	0	4	0	0	0	0	0	0	1
3	0	0	14	0	0	0	0	0	0	0	0	1	0	0	0	2
4	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	3
5	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0
6	2	2	0	0	0	12	0	0	0	0	0	0	0	0	0	4
7	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	1
8	0	3	0	0	0	0	0	18	0	0	0	0	0	0	0	10
9	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	1
10	0	1	0	0	1	0	0	0	1	16	2	0	0	0	0	12
11	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0
12	0	0	0	1	0	0	0	0	2	0	0	59	0	0	0	3
13	0	0	0	0	0	0	0	0	0	0	0	0	32	0	0	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	16
15	0	1	0	0	0	0	0	0	0	0	0	0	0	0	44	4
16	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	19

Table 4.7: Confusion matrix for Translation versus Roll. The average precision is 87% with a minimum of 24% on class 16. The average recall is 79% with the lowest of 27% on class 14. In table 4.9 the lowest precision is observed for class 5.

Actual ↓	Prediction → 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	54	0	0	0	0	0	0	0	0	0	0	0	0	1	0
3	0	0	24	0	0	0	0	0	0	0	2	0	0	0	0	1
4	0	0	0	24	0	0	0	0	0	0	0	1	0	0	0	2
5	0	0	0	0	44	0	0	0	0	0	0	0	0	0	0	1
6	0	0	0	0	0	24	0	0	0	0	3	0	0	0	1	5
7	2	0	1	0	1	0	16	0	0	0	0	0	0	0	0	3
8	0	3	0	0	2	0	0	34	7	0	1	0	3	0	1	1
9	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0
10	0	0	0	6	0	0	0	0	0	29	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	33	0	0	0	0	0
12	0	0	1	0	0	0	0	0	1	0	5	26	0	0	0	10
13	0	0	0	0	0	0	0	0	2	0	0	0	31	0	1	6
14	0	0	0	0	0	0	0	0	0	0	0	0	0	15	2	6
15	0	1	0	0	0	0	0	0	0	0	0	0	0	0	36	0
16	1	0	0	0	3	0	0	0	0	0	0	1	0	1	0	9

Table 4.8: Confusion matrix for Translation versus Expressions.

The main difference between Z-Translation and all the other sets, except Freestyle, is that the distance to

the sensor was kept fixed at 0.6 meters from the sensor, while for Translation the pose was frontal and the distance to the sensor was varied between 0.4 to 1.2 meters. We remind the reader that the resolution decreases as distance to sensor increases. This is an indication that the optimum resolution (not the scale — the camera observations are scale invariant) for the SHOT descriptor is somewhere above 0.6 meters, where it reaches optimum descriptive power.

Actual ↓	Prediction → 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	6	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	10	0	0	0	0	1	0	1	0	0	1	4
3	0	0	15	0	2	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	7	4	0	0	0	0	1	0	0	0	0	1	0
5	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	13	7	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	2	0	5	0	0	0	0	0	0	0	0	2
8	0	0	0	0	3	0	0	26	1	1	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
10	0	0	0	0	2	0	0	0	0	28	0	1	1	0	1	0
11	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	65	0	0	0	0
13	0	0	0	0	1	0	0	0	0	0	0	0	32	0	0	0
14	0	0	0	0	3	1	0	1	2	1	0	1	2	9	0	2
15	0	0	0	0	2	0	0	0	0	0	0	6	0	0	41	0
16	0	0	0	0	4	0	0	0	1	3	1	1	0	0	1	9

Table 4.9: Confusion matrix for Yaw versus Roll. We can observe that class 5 (ben_w) with a total of 61 camera observations for Yaw (training) and 30 for Roll (testing) has a high number of false positives. Furthemore, the recall for class 5 is one. Another interesting observation is that class 2 has zero precision, with no correct predictions. The average precision is 80% and average recall is 68%.

Actual ↓	Prediction → 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	13	0	0	0	3	0	2	0	0	0	0	0	0	0	0	0
2	0	14	0	0	0	0	0	0	0	1	0	0	0	0	2	0
3	0	2	14	0	0	0	0	0	0	0	0	0	0	0	1	0
4	0	0	0	5	2	0	0	0	0	0	0	0	0	0	6	0
5	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0
8	0	10	0	0	1	0	0	20	0	0	0	0	0	0	0	0
9	0	5	0	0	0	0	0	0	8	0	0	0	0	0	0	0
10	0	9	1	0	0	0	0	2	0	20	0	0	1	0	0	0
11	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0
12	0	1	0	0	0	0	0	0	0	0	0	48	0	0	16	0
13	0	0	0	0	2	0	1	0	0	0	0	0	29	0	1	0
14	0	12	0	0	2	0	0	0	2	0	0	0	1	3	2	0
15	0	0	0	0	0	0	0	0	0	0	0	1	0	0	48	0
16	0	3	1	0	0	0	1	0	0	0	1	0	2	0	0	12

Table 4.10: Confusion matrix for Pitch versus Roll. Compared to table 4.9 the results are more balanced. Here, class 2 has 25% precision while classes 1, 4, 6, 14 and 16 have 100% precision. Furthermore clases 5, 6, 7 and 11 have 100% recall, with the lowest of 13% for class 14. The average precision is 85% and average recall is 74%.

As further support, we remind the reader that the SHOT feature descriptor was designed for object recognition, possibly optimized for scenes, where objects are usually further away. Moreover, both Roll and Expressions have frontal poses. If we compare this to Roll vs Expressions (row 3, column 5), we observe that the accuracy is not as high: if the distance to the sensor would not make any difference then we would see similar results. Since this is not the case, then it must be the distance to the sensor which causes a higher accuracy for when Translation is used for training.

Furthermore, since Translation versus Expressions is slightly an outlier on the 5th column, it could be the

case that during the recording procedure of the Expressions dataset, the distance to the sensor was not kept fixed for all subjects. Then, if the optimum descriptive power for SHOT is at a distance higher than 0.6 meters and if some variance in distance to sensor is present in the Expressions dataset, this would result in the absolute highest accuracy recorded.

Even if we disregard the previous premise of measurement inconsistency for the Expressions set, we can still observe that the highest accuracy recorded when Roll is used for training is still on testing on the Translation dataset (3rd row, 4th column) and the same can be said for Expressions (5th row, 4th column), which is intuitive since frontal poses are less difficult to classify. In other words, the maximum on the rows for Roll and Expressions, both frontal, is still while testing on Translation.

The next highest accuracies are equal and were recorded for when the Yaw and Pitch is used for training and Roll is used for testing. The confusion matrices are given for both results in tables 4.9 and where the class indexes correspond to the alphabetical sorting of the class names in table 2.1 after removing the ones with the "glasses" suffix.

However, if we observe all the accuracies for row 1 (Yaw is used for training), we can clearly see that testing on Roll is an outlier. Furthemore, we can observe a slightly symmetric relationship between 2nd row, columns 3, 4 and 5 and column 2, rows 3, 4 and 5. The similarity between Roll, Translation and Expressions is that they are all frontal.

When training on Pitch, we have good results on frontal pose sets, which is intuitive since pitch also contains *some* frontal poses (the subjects were asked to look up and down). This is an indication that the best performances are also achieved when there are enough correlations between feature vectors extracted in similar poses. Furthermore, the same can be said for the next inner rows and columns, namely row 3, columns 4 and 5, and column 3, rows 4 and 5.

In general when the Freestyle set was used for training, the lowest accuracies were recorded (last row, 6th column) since it contains all types of variances and is not pose normalized. Then, when Freestyle is used as the training set, the average accuracy (last column, 6th row) is also the lowest.

We will now focus on the first three cells in the 6th column, where Freestyle is used for training and testing is done on Yaw, Pitch and Roll. An interesting observation which is contrary to the cross validation results is that when Freestyle is used for training, the highest accuracy is not recorded on the Roll dataset. Instead, we can see that the highest accuracy is recorded for Yaw. If Freestyle would have been pose normalized, then we would expect these three accuracies to be similar, however, this is not true.

Then, the only explanation for the similarities between Freestyle and Yaw is that the latter dataset was the most difficult to pose normalize and as such it has the highest pose correspondence to Freestyle between Roll, Pitch and Yaw testing sets. The confusion matrices for these three cases are given in tables 4.11, 4.12 and 4.13. In all three tables we can observe that column 8 has a low precision.

The pose normalization step was overtrained on class 8, hence the best results of the normalization process are obtained for that class. hen looking at column 8 in the three confusion matrices, we can observe that the class with the lowest precision is class 8 since the observations of this class are pose normalized accurately and freestyle contains observations without normalization, hence the low precision.

Conversely, if we look at the confusion matrix of the best result in table 4.8 we see 100% precision for class 8. Hence, we can conclude that the accuracy is high when either there is no pose variance or when the data is not pose normalized. Since the observations are not labeled with the angles for yaw pitch and roll we are not able to analyze this issue in further detail.

Actual	Prediction →															
↓	1	2	3	4	5	6	7	**8**	9	10	11	12	13	14	15	16
1	16	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2
2	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	4	0	11	0	3	0	0	0	0	0	0	1	0	3	0	7
4	0	0	0	9	4	0	0	2	0	0	2	0	0	1	0	0
5	1	0	0	0	59	0	0	0	0	0	0	0	0	0	0	1
6	1	0	0	0	2	6	0	5	0	0	1	0	0	4	0	7
7	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	7
8	0	0	0	0	0	0	0	38	1	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	2	24	0	0	0	0	5	0	1
10	0	0	0	0	0	0	0	32	1	13	3	0	0	9	0	5
11	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0
12	0	0	0	0	0	0	0	1	3	0	2	41	0	4	4	8
13	0	0	0	0	0	0	0	27	0	0	0	0	2	4	0	0
14	0	0	0	0	0	0	0	4	0	0	1	0	0	17	1	1
15	0	0	1	0	0	0	0	7	0	0	2	5	0	5	20	6
16	2	0	0	0	0	0	0	5	0	0	2	0	0	5	0	18

Table 4.11: Confusion matrix for Freestyle versus Yaw. Class 8 has low precision.

Actual	Prediction →															
↓	1	2	3	4	5	6	7	**8**	9	10	11	12	13	14	15	16
1	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	3	0	0	0	0	1	1	25	5	0	0	0	0	3	0	5
3	0	5	4	0	0	0	0	6	0	0	2	34	0	0	1	0
4	0	1	0	3	3	0	0	5	1	0	0	2	0	0	0	0
5	1	11	0	0	36	0	0	6	0	2	1	0	0	2	0	0
6	5	11	0	0	3	7	0	4	0	0	2	1	0	0	0	0
7	2	2	0	0	0	0	16	1	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	42	0	0	0	0	0	0	0	0
9	0	1	0	0	0	0	0	1	19	0	0	1	0	0	0	0
10	1	0	0	0	1	0	0	13	0	8	0	3	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0
12	1	0	0	0	0	0	0	3	0	0	2	40	0	0	0	0
13	0	10	0	0	0	0	0	26	0	0	2	0	1	0	0	0
14	0	2	0	0	0	0	0	1	0	0	0	0	0	10	0	0
15	0	5	0	2	3	0	0	0	0	0	2	24	0	2	24	0
16	3	2	0	0	0	0	0	14	0	1	1	8	0	0	0	0

Table 4.12: Confusion matrix for Freestyle versus Pitch. Low precision for class 8 and class 12.

Actual	Prediction →															
↓	1	2	3	4	5	6	7	**8**	9	10	11	12	13	14	15	16
1	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	2	0	0	10	0	0	0	5	0	0	0	0
3	8	0	0	0	3	0	1	0	0	0	1	4	0	0	0	0
4	0	0	0	8	2	0	0	2	0	0	1	0	0	0	0	0
5	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0
6	1	0	0	0	3	5	0	6	0	0	1	3	0	1	0	0
7	0	0	0	0	0	0	7	0	0	0	2	0	0	0	0	0
8	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	4	8	0	0	0	0	1	0	0
10	1	0	0	0	1	0	0	21	1	7	0	0	0	1	1	0
11	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0
12	0	0	0	0	0	0	0	0	2	0	0	60	0	3	0	0
13	0	1	0	0	0	0	0	29	0	0	1	0	2	0	0	0
14	0	0	0	0	0	0	0	4	0	0	0	0	0	18	0	0
15	1	0	0	0	15	0	0	0	0	1	0	23	0	6	3	0
16	0	0	0	0	1	0	0	17	1	0	0	0	0	0	0	1

Table 4.13: Confusion matrix for Freestyle versus Roll. Low precision for class 8, zero precision for classes 2 and 3.

Correlation distance

The sample correlation distance (eq. 4.1) is a linear metric which is derived from the sample variance and covariance between two vectors and is a measure of multivariate dependence. It is zero if the vectors are statistically independent. This metric is often interpreted as an energy measure between two probability distributions.

$$d_{corr}(A,\ B) = 1 - \frac{(a_i - \overline{a})\ (b_i - \overline{b})'}{\sqrt{(a_i - \overline{a})\ (a_i - \overline{a})'}\ \sqrt{(b_i - \overline{b})\ (b_i - \overline{b})'}} \tag{4.1}$$

If the data is standardized, then the vectors will each have a mean of 0 and a standard deviation of 1, hence the correlation distance becomes the square euclidean distance divided by twice the dimensionality of one vector — given that the vectors have the same dimension:

$$d_{corr}(A,\ B) = 1 - \frac{l_2^2(A,B)}{2n} \tag{4.2}$$

The distance function is applied as a metric for the KNN and the results are displayed in table 4.14.

Testing → Training ↓ N =	Yaw 26722	Pitch 29632	Roll 20707	Z-Translation 41093	Expressions 28774	Freestyle 61294	Average Train
Yaw	-	0.4634	0.7800	0.6160	0.5413	0.5097	0.5821
Pitch	0.3490	-	0.7653	0.7173	0.7413	0.3241	0.5794
Roll	0.4953	0.6520	-	0.6840	0.6716	0.3549	0.5716
Z-Translation	0.3959	0.6593	0.7971	-	**0.8404**	0.3493	0.6084
Expressions	0.4146	0.6355	0.6308	0.6827	-	*0.3160*	0.5359
Freestyle	0.6023	0.4762	0.5355	0.5213	0.6023	-	0.5475
Avg. Test	0.4514	0.5773	0.7017	0.6443	0.6794	0.3708	0.5708

Table 4.14: Sum rule over one KNN classifier with $K = 3$ is trained using the correlation distance. The overall accuracy is higher than when using the Euclidean distance.

The overall accuracy is about 1% higher than when using the euclidean distance, however we can observe that in two cases — Translation vs Freestyle and Translation vs Roll — the accuracy is in fact lower. Overall the same patterns as before are observable. If we disregard the columns and rows for Yaw and Freestyle, we can observe that there is a slight simmetry between the results, as in table 4.6. If we focus on the outer rows and columns, we can observe that the highest result for Freestyle both when training and testing is obtained on Yaw, which implies high correlation between the two datasets. As discussed earlier, the similarity consists in the poor pose normalization of Yaw and the fact that Freestyle is not pose normalized at all. Roll, Translation and Expressions are captured using frontal poses with the pose normalization being the best on Roll. Pitch is partly in the former group since either the pose normalization was either inconsistent or there were camera frames which did not require pose normalization, since the face was already in frontal pose. At this point, we can conclude that there are 3 groups according to the degree of pose normalization: the frontal pose groups composed of Roll, Translation and Expressions, the poorly pose normalized Yaw and Pitch and Freestyle which is not pose normalized at all.

4.6.2 Sum rule and ensembles

The correlation distance was kept as the default metric and also used for K-means clustering. The results are displayed in table 4.15 where the overall accuracy is lower than when using one single classifier architecture. There are however two cases when the accuracy is higher than the results in the previous table, 4.14, when training on Pitch and testing on Yaw and when training on Pitch and testing on Roll we can observe a 1% increase in accuracy. Also there is an increase, however not as significant when training on Pitch and testing on Yaw.

The overall lower accuracy is due to the fact that the features and the metric used do not cause the unsupervised identification of face regions to be optimal, there is not enough diversity generated between each cluster. In other words, the clusters do not represent face regions and instead the feature vectors are clustered based on other criteria, even though the data used for clustering contains both the feature vector and the XYZ position of the SIFT keypoint — which represents the topological position of the extraction point.

Testing → Training ↓ N=	Yaw 26722	Pitch 29632	Roll 20707	Z-Translation 41093	Expressions 28774	Freestyle 61294	Average Train
Yaw	-	0.4487	0.7677	0.5960	0.5413	0.5073	0.5722
Pitch	0.3565	-	0.7726	0.7147	0.7248	*0.3201*	0.5777
Roll	0.4991	0.6447	-	0.6773	0.6495	0.3282	0.5598
Z-Translation	0.3846	0.6447	0.7946	-	**0.8385**	0.3468	0.6018
Expressions	0.4128	0.6044	0.6161	0.6733	-	*0.3201*	0.5253
Freestyle	0.5685	0.4597	0.5183	0.5147	0.4495	-	0.5021
Avg. Test	0.4443	0.5604	0.6939	0.6352	0.6407	0.3645	0.5565

Table 4.15: Ensembles and Sum rule accuracy over pair test using K=3 neighbors and M=50 clusters. The correlation distance was used for both K-means and the 50 KNNs. For an overview of the algorithm, please refer to fig. 3.9 on page 38. The sampling method is based on SIFT keypoints instead of uniform sampling and patches as in fig. 3.9.

4.6.3 Combining MLPs with KNN experts

In this series of experiments we consider an application where the KNNs have already been trained using some data and MLPs are continuously trained to correct the mistakes of the KNNs. We consider the situation where the identity of one or more subjects is already correctly estimated. If the subject is continuously in front of the camera — e.g. being tracked — then we can use this ground truth to train an MLP. This application would be feasible only when there would be a small number of fixed classes since we are using the outputs of the KNNs as input for the MLPs. Hence, these results are not necessarily correlated with the previous results since we are using the ground truth from the test set to train the MLPs.

In the previous method, the final decision was simply based on the class with the largest support, from the combined posterior probabilities of each expert using the sum rule. In this section, two new methods are proposed, where MLPs are combined with KNNs towards improving the classification performance. There are two stages in this configuration of ensembles. First, the KNN experts are trained as usual using one of the recorded datasets in table 2.1 using the method depicted in figure 3.9, on the right. During the testing phase, two MLPs are trained online separately, based on the outputs of the KNN experts. This is advantageous since it allows us to adapt the final decision while observing new data.

4.6.4 Stochastic gradient descent

MLPs usually consist of three or more layers of neurons. In our case we use only one hidden layer of 50 hidden neurons. The number of inputs are the same for both MLPs, namely the posterior probabilities (the outputs) for each expert are sorted according to the index of the expert and concatenated in one vector of dimension $E \times \Omega = n$. Since we have a dataset with 16 classes and 50 experts, we have a total of 800 inputs for both neural networks. The number of outputs is however different. In one case we have a vector equal to the number of classes, where the class with the largest support wins and in the other case we have 50 outputs which represent the weights for each expert. The activation function we use is the hyperbolic tangent function (eq. 4.4), and as such the inputs and outputs are rescaled to accommodate to this activation function (we have values between [0, 1] from the KNN experts).

There are two basic steps involved in the online training of the neural networks using back-propagation. The first stage is the forward propagation of the signal through the network. During the forward pass, each hidden neuron computes the weighted sum of its inputs to form its net activation where w_{j0} corresponds to the bias unit for $x_0 = 1$ as in eq. 4.3. Then, the activation of h_j is computed using the activation function, in our case the hyperbolic tangent function 4.4.

$$h_j = \sum_{i=1}^{n} x_i w_{ji} + w_{j0} \tag{4.3}$$

$$f(h_j) = tanh(h_j) \tag{4.4}$$

In eq. 4.3 i represents one input node and thus indexes the lower input layer and j the upper hidden layer. w_{ji} is the weight from the lower neuron i to an upper neuron j and is the equivalent to a biological synapse. Then, the same procedure is applied to the next layers, until the output neurons. Finally, the activation is applied to all the output units as well. The outputs are effectively computing discriminant functions. We can therefore write the complete forward pass to one output neuron as in eq. 4.5. The output neurons are indexed using the letter k, where n_H is the number of neurons in the hidden layer.

$$z_k = f\left(\sum_{j=1}^{n_H} w_{kj}\, f\left(\sum_{i=1}^{n} w_{ji} x_i + w_{j0}\right) + w_{k0}\right) \tag{4.5}$$

After a pattern has been presented to the network and activations for all outputs have been computed, the difference between the output and the target is computed for each neuron and this value represents the error at each neuron. As an indication of the total error, our objective is to minimize a singular function criterion. The Mean Squared Error (MSE) is a common quadratic loss function, which measures the average of the squares of the errors (eq. 4.6). The targets for each output (ground truth) is denoted by the letter t, where the total number of outputs is n_o.

$$MSE = \frac{1}{n_o} \sum_{k=1}^{n_o} (t_k - z_k)^2 \tag{4.6}$$

For the first network the targets are a vector of equal length to the number of classes, where the correct class is denoted by 1 and the rest are zeros, obviously there can be only one correct class. However, in this way similarities could be learned by using non-discrete values for the ouputs. In the second case, where we intend to learn the weights for each expert, we have a number of outputs equal to the number of experts. In this case the target vector is represented by a binary vector where if the expert k has made a correct classification the target is set to $t_k = 1$ and an incorrect classification is set to zero.

The back-propagation of error (adapting the weights) is based on gradient descent. Initially the weights are initialized to random values. After one pattern has been presented to the network, the MSE is computed and the weights are adjusted in the direction that will minimize the MSE. Since we are presenting

the examples in random order, the process is called stochastic gradient descent. Thus, as each example is presented to the network (time step T), the weights are adjusted in proportion to the rate of change in MSE as in eq. 4.7.

$$W(T+1) = W(T) - \eta \frac{\partial MSE}{\partial W(T)} \tag{4.7}$$

Since the error signal for internal neurons are not known, it is impossible to directly compute the error signal δ for internal neurons. As such, this is done by propagating back the error signal for each neuron, according to eq. 4.8. This is effectively equivalent to performing the forward pass in reverse, using the difference between the outputs and the target as input. Thus, δ is computed for each neuron, according to the weights, layer by layer, starting from the output layer, in which case the equation reduces to the simple difference between targets and outputs: $\delta = t_k - z_k$. In this notation δ_{in} is the upper layer, δ_{out} is the lower layer, n_{out} is the number of neurons in the lower layer. For the bias unit in the input layer, the same procedure is applied.

$$\delta_{in} = \sum_{out=1}^{n_{out}} f'(h_{out})\, \delta_{out}\, w_{in-out} \tag{4.8}$$

When the error signal has been computed for all neurons, then the weights are updated, starting from the input layer, using eq. 4.9, where w_{ji} is the weight connecting input neuron i to hidden neuron j, δ_j is the error at hidden neuron j and f' is the derivative of the activation function.

$$w_{ji}(T+1) = w_{ji}(T) + \eta\, \delta_j\, x_i \tag{4.9}$$

Although the error usually decreases after most weight changes, there may be derivatives that cause the error to increase as well. Unless learning rates are very small, the weight vector tends to jump about the error function. We used a learning rate $\eta = 0.9$ which implies that the MLPs have a very short term memory in the weight space. This causes the network to give more importance to what it has seen more recently. The magnitudes of the jumps are proportional to the learning rate η. During online training it is less likely for an MLP to get stuck in a local minima, since the weight values change in a more chaotic way. Furthermore, if there is a high degree of redundancy in the training data, such as in our case, the online mode is superior to the batch mode since the weights will be updated more often in one given iteration, while in batch training, it will take longer to evaluate one training step.

Even though the MLPs are trained online, we can think of the MLP training data as a new dataset $\mathcal{D}$, containing a number of observations N' equal with the number of probes in the testing set. Instead of summing the outputs of E KNN classifiers, we average the output. In [Kittler et al., 1998] the authors conclude that there is no essential difference between summing and averaging when combining classifiers. This is intuitive since the support from each expert is relative to the numerical limits of the outputs, regardless of summing or averaging. In the case where an expert does not output any posterior probabilities — since there are no feature vectors from that particular cluster — we consider that the expert has generated a zero posterior probability vector.

An important aspect is that the ground truth for the online training phase of the MLPs is already known, since the training set is also labeled with the true class. However, for each vector x, the training step is performed *after* the current vector x passed to the neural network and the outputs are computed. As such, the final classification decision in both cases is based on the output of the network, before a back-propagation training step has been performed with that particular example. As such, the weights of the network are unaffected by the current example.

4.6.5 Dynamic weights for experts

The idea behind weight space learning is to constantly change the weights for each expert $C_{j...M}$ by considering the input space $\mathcal{D}'_w$ as one multidimensional signal, equal to the number of clusters E times the number of classes Ω. Then, during testing, the weights for the decisions of each expert can be adjusted according to the pattern of the support given by the experts, per probe camera observation $\mathcal{O}$. As such, MLP_w is then trained online to increase or decrease the weights for each expert. In this case, the final decision is based on a dynamically weighted sum rule.

Testing → Training ↓	Yaw 26722	Pitch 29632	Roll 20707	Z-Translation 41093	Expressions 28774	Freestyle 61294	Average Train
Yaw	-	0.5220	0.7408	0.5987	0.5578	0.5081	0.5855
Pitch	0.4015	-	0.7751	0.7227	0.7431	*0.3541*	0.5993
Roll	0.5272	0.6740	-	0.7027	0.6862	0.3630	0.5906
Z-Translation	0.4165	0.7125	0.8240	-	**0.8514**	0.3639	0.6337
Expressions	0.4278	0.6740	0.6333	0.6987	-	0.3614	0.5590
Freestyle	0.5816	0.4835	0.5232	0.5347	0.4734	-	0.5193
Avg. Test	0.4709	0.6132	0.6993	0.6515	0.6624	0.3901	0.5812

Table 4.16: Cluster weights accuracy over pair tests. The average accuracy is higher than in all the previous methods. The same patterns as discussed above are present.

As we can see from the results in table 4.16, the overall accuracy is higher than all the previous methods, with the highest accuracy recorded on the z-translation set used as training and roll and expressions as test sets. The lowest accuracy, similar to previous results is with the pitch set used for training and expressions for testing. This is the case since pitch contains frames which have either the top or bottom part of the face missing (due to sensor recording position), hence the features are representative for top or bottom views, not frontal views such as for the expression set. The lowest accuracy is no longer observed between Expressions and Freestyle but for Pitch vs Freestyle.

4.6.6 Product rule

The next strategy was to stack [Wolpert, 1992] MLP_p after the decision of the KNN classifiers, thus replacing the sum rule with another classifier. However, testing this method resulted in a lower accuracy of approximately 10% on two pair tests. Instead of having the final decision based only on the outputs of MLP_p, the two methods are combined using the product rule. The results of the sum rule $\Phi^{\sum}_{i...\Omega}$ for each class ω of the KNN experts is combined with the output $\Phi^{MLP}_{i...\Omega}$ of the online trained MLP_p, using the product rule as in eq. 4.10, where the class with the largest support represents the final decision. The results are displayed in table 4.17.

$$\Omega(\mathcal{O}) = \arg\max_{c} \left(\Phi^{\sum}_{c}(\mathcal{O}) \; \Phi^{MLP}_{c}(\mathcal{O}') \right) \text{ where } \mathcal{O}' \in \mathcal{D}'_s \tag{4.10}$$

Since the MLP is trained online, it has the ability to correct the decision of the KNN experts and as such higher accuracies are observed. Even though the methods using MLPs result in higher accuracies they are not practical since when adding a new class the number of inputs for the MLP has to change. However, this proves that another classifier can be used to further improve accuracy.

Testing → Training ↓	Yaw 26722	Pitch 29632	Roll 20707	Z-Translation 41093	Expressions 28774	Freestyle 61294	Average Train
Yaw	-	0.6758	0.7677	0.7427	0.6881	0.6410	0.7031
Pitch	0.5572	-	0.7873	0.8053	0.7872	0.5389	0.6952
Roll	0.5816	0.7491	-	0.7800	0.7450	*0.5073*	0.6726
Z-Translation	0.5460	0.7894	**0.8386**	-	0.8330	0.5186	0.7051
Expressions	0.5141	0.7399	0.7482	0.8187	-	0.5113	0.6664
Freestyle	0.6998	0.6630	0.6210	0.6987	0.6239	-	0.6613
Avg. Test	0.5797	0.7234	0.7526	0.7691	0.7354	0.5434	0.6839

Table 4.17: The sum decision of the KNN experts is combined using the product rule with the decision of a MLP trained online. An average increase in accuracy of 10% can be observed, when compared to the average result in table 4.16

4.7 Knowledge transfer

In this experiment the Pitch, Roll, Translation and Expression datasets were combined into training set A, the Yaw was kept to pre-train the MLP as set B and the Freestyle was used for testing as set C. Two tests are performed in order to be able to compare the results with the previous ones. In the first case, the ensembles of KNNs are trained using set $A + B$ and the testing is done on set C. The recorded accuracy was 0.6361 ± 0.0353 with 99% confidence. With the second architecture, the ensemble of KNNs is trained on A, the ensembles are tested on B, then the MLPs are trained in batch mode based on the outputs of the KNNs while testing on B and finally using the combination of KNNs and MLPs as described above (the weights and the product rule) to test on C. The results are given in the table 4.18 below:

Architecture	Normal	Softmax
KNN	0.6361 ± 0.0353	-
$KNN + MLP_p$	0.5405 ± 0.0366	0.5219 ± 0.0367
$KNN + MLP_w$	0.4052 ± 0.0361	0.4773 ± 0.0367

Table 4.18: Knowledge transfer results. The results are better than the best obtained in table 4.18.

As we can see, the KNN ensemble architecture, in the first row of table 4.18 is higher than the best result (50%) obtained while training on Yaw and testing on Freestyle (table 4.14). Furthermore, the results in the second column are also higher than this result. The average testing on Freestyle was ~36% while here the minimum is 40% accuracy.

4.8 All-vs-All SVMs

The performance of an all-versus-all multi-class SVM architecture using a radial basis function kernel was evaluated during 5 fold cross-validation runs on a multi core machine. The training was done using Sequential Minimal Optimization (SMO) for which the Karush-Kuhn-Tucker (KKT) violation level was set to 0.05 during a maximum iteration of 15000 epochs. Directed acyclic graphs (DAG) were not used while testing the pair results.

The average cross-validation error was used as the objective measure for minimizing the error. First, a grid search with the box constraint C and $\sigma \in \{0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000\}$ was performed. The parameters found during the grid search were then used as the starting search interval $\pm 5\%$ for

fine tuning the search using Particle Swarm Optimization (PSO). The highest accuracy of 0.8278 was registered for the Pitch dataset when the parameters were $C = 871.7$ and $\sigma = 444.7$ and 0.8651 accuracy on the Roll dataset where $C = 900.6$ and $\sigma = 409.7$. These two results are lower than the previous cross-validation result. These results indicate that the decision boundary is so complex that a large percentage of the data points were used as support vectors.

Chapter 5

Discussion

5.1 Summary

In this research the potential of using low resolution, off-the-shelf sensors for 3D face recognition in the presence of occlusions due to pose variance and expression was investigated. Three object recognition feature descriptors were selected based on performance and computational complexity and evaluated in the context of a non-holistic, atomistic representation.

Two lazy learning architectures, based on the K-nearest neighbor classification model, were used to approximate the target function locally for each face region or for the complete face, in order to capture the dynamic shape cues and achieve robustness towards occlusions. The first method consisted of a single KNN classifier, which was trained using the feature vectors extracted per each camera observation.

For the second architecture, an ensemble of KNN experts were used in order to simplify the decision boundary. Two unsupervised methods, namely K-means and Self-Organizing Maps were used to identify face regions, on four different scales. The results of both methods were combined using either majority voting or the sum rule, where the class with the largest support was used as the final classification decision.

Two sampling methods were evaluated: uniform sampling on a small scale experiment using an 8 class dataset with no pose variance, where the sampling resolution and the number of clusters was varied for all three feature descriptors. The best performing feature descriptor for the current architectures was identified, while the optimal context of using the other descriptors was also suggested.

A novel sampling method based on the SIFT algorithm was used to sample data economically and consistently, around different face regions with high degree of curvature, similar to a sliding window sampling method. This pose invariant sampling method was used to extract a total of $N = 200K$ feature vectors of dimensionality 352 using the SHOT descriptor from an 18 class dataset consisting of 4675 camera observations.

A real-time head tracker was used to detect the head and crop the face. This process was augmented with a continuous filtering step which was developed to filter out artifacts such as long hair and other disconnected components based on clustering 3D points. A further pose normalization prepossessing step using ICP was subsequently used as a continuous step after the detection and filtering processes in order to solve the left-right translation due to extreme yaw pose angles which also partly solved the rotation variance. This method showed promising results for roll movements of the head, with angles

ranging in the interval of $\pm 45^{\circ}$. The rotation variance of camera observations originating form left-right movements (yaw) of the head was however not consistently solved.

The dataset was recorded in real time and consists of six subsets in order to observe the impact of every type of variance: pose (yaw, pitch, roll), distance to sensor, expressions and a final one where the final pose normalization step was not applied, containing all types of variances.

The complete architecture is completely automatic and requires no supervision from humans, except for the class labels of the subjects. Due to the lazy learning models, the architecture is also scalable in the number of classes and has the potential to be applied to real-time applications since high computational complexity was avoided during all phases of development.

5.2 Conclusions and future work

The proposed, atomistic architecture was able to correctly identify 8 subjects, with maximum accuracies going up to 99%, and was robust towards missing data in the context of no pose variance. This suggests that this methodology can be applied to biometric authentication, where the subjects usually go through a standard registration procedure which does not typically involve pose variance.

We estimate that the number of classes that can be correctly classified decreases with the distance to the sensor since this decreases resolution. Furthermore, the right scale and resolution depends entirely on the feature descriptor, as we have seen in the pair tests, the SHOT feature descriptor is optimal at a lower distance to the sensor. Furthermore, the number of sampled keypoints should also depend on the type of feature descriptor, which also influences the number of clusters and in turn the number of experts.

We have shown that it is possible to identify face regions in an unsupervised manner. However, the accuracy of the ensembles in such an architecture depends critically on the diversity of data used to train each expert, which is in effect a consequence of the clustering outcome. The two clustering algorithms showed similar results, for the K-means clustering the process was repeated 10 fold and the solution with the best separation was kept, in all cases. Self-organizing maps could be further augmented to dynamically learn weights between clusters, as a function of the feature vectors, in a way that would measure the topological agreement between the features during testing time and thus change the confidence according to the "agreement" between adjacent face regions.

It was the case that the simple architecture performed 2 to 3% better in all cases compared to the ensembles, mainly due to clustering which generated smaller training sets which were not disjoint enough. The correlation distance showed an improvement of 2% accuracy for both methods. However on the large scale dataset it was clear that the limits of this metric and feature descriptor was reached since noisy training data increases the complexity of the decision boundary unnecessarily, since the abstraction made during the training phase was not adequate.

The only possibility of further increasing accuracy with this metric and the SHOT feature descriptor would then be to simplify the decision boundary by improving the ensemble method. An attempt to further increase the clustering performance and thus ensemble performance was made by concatenating the pose normalized topological position of the extraction point and the feature vector which resulted in increased accuracy, depending on the type of pose variance. We have however shown that it is possible to sample consistently even from pose-variant data using SIFT as a viewpoint invariant sampling method, based on surface curvature.

The average cross-validation accuracy for the whole dataset was 95% $\pm 2\% \times 10^{-2}$ with 99% confidence. The 16 class dataset consisted of $N = 208222$ observations of dimensionality 352. The highest accuracy was recorded for the Roll dataset ~97% which was the most accurately pose normalized during the

recording procedure, while the lowest was recorded for Freestyle ~90% which was not pose normalized at all.

However, when putting the architectures to the tough test of training on examples containing a specific type of variance and testing on a dataset with different variance, we have registered lower accuracies. In the process we have identified three groups of variances: the Roll, Translation (distance to sensor) and Expressions set have highly correlated and almost symmetric results since the pose is frontal in all cases. Freestyle is the most difficult set since it contains frames with extreme angles, expressions varying distance to sensor, etc and is the second group. In this spectrum, Yaw is somewhere between the two groups, but however closer to Freestyle, while Pitch is closer towards the first group.

These observations lead us to the conclusion that "premature optimization is the root of all evil" (Donald Knuth). Instead of having some of the data pose normalized and some of the data not pose normalized a problem was created instead of solving one. Given that the SHOT feature descriptor is not rotation invariant and knowing that the sampling method is consistent regardless of pose, then the goal should be to compute feature vectors in the original pose. We have neither focused on solving the problem of normalization nor completely ignoring the problem.

A dataset which contains feature vectors computed at all types of poses, given the consistent sampling method would have enough examples from each pairs of face regions for all poses, if the pose normalization step would be completely skipped. However, this would require a stronger feature descriptor and a metric which can accurately discriminate the extracted feature vectors. Then the pose normalization for the SIFT keypoints would be dropped as well. Having the topological position of each keypoint detected by SIFT, semi-supervised could be applied to increase accuracy.

Furthermore, hierarchical clustering could be used in order to establish generic face regions (e.g. eyes, mouth, nose, etc), then having subdivisions for each lower clusters in the dendrogram, based on the pose variance and expressions. This method would generate clusters with enough diversity which would in turn simplify the decision boundary since there would be an expert not only for each face region, but also for each face region per pose. This hierarchical approach would facilitate the previous idea to correlate the agreement between the face regions, according to the position of the query feature vectors in the dendrogram.

The highest accuracies were observed when the Z-Translation dataset was used for training, where the distance to the sensor was varied between 0.4 ant 1.4 meters, which suggests that the SHOT feature descriptor does not describe surfaces optimally at 0.6 meters — where mostly all sets were recorded. Then this implies that the feature descriptor is not invariant to resolution since the scale is kept the same, even when distance to sensor increases. Since the feature descriptor was designed for object recognition, this further supports the idea that SHOT is better at describing surfaces which are at a distance larger than 0.6 meters. This, of course, influences the clustering results as well.

During the K-means clustering there is no guarantee that the feature vectors used for training will converge towards centroids based on the topology of the face. In other words, the feature vectors are grouped based on different criteria and thus can not differentiate a partial observation of a nose from a partial observation of an eye, for example. This type of noisy training data increases the complexity of the decision boundary unnecessarily, since the abstraction is ambiguous. Since all sets contain feature vectors which are sampled in variable positions, then a sampled keypoint will not always land on the tip of the nose for example and as such we would expect that the distance to a centroid would explain similarity between a feature vector to a part of the face (e.g. "how much does this part look like an eye").

We therefore believe that a stronger feature descriptor is required and we propose the use of a uniform binary cube mask with a to-be-determined resolution and size which would simply measure the presence of a point in space and mark a presence with 1 or 0 an absence. Concatenating these results in a vector starting from the XY plane and moving on to the Z plane would result in high dimensionality sparse

vectors which could then be used within the same architecture in combination with the simpler and faster Hamming distance.

We have identified during the small scale experiment that the ESF feature descriptor is a good candidate for an initial weak rejector which could be used on the complete camera frame in order to increase the recall. Similarly, the PFH descriptor could be used to densely sample with low radius in the context of a Bag of Features method [Li et al., 2010] in order to construct a vocabulary of face regions which casts votes. Since each type of pose variance generates a unique type of occlusion / missing data, as observable in fig. 4.1, the code-book for each camera frame could be used to train specialized ESF whole frame classifiers as the 2nd tier classifier.

Furthermore, nonlinear metric learning [Xing et al., 2002], kernel spectral clustering, tensor kernels for SVMs and generative models are some of the few ideas that could generate truly scalable and compact representation of the input space.

Bibliography

[Abate et al., 2007] Abate, A. F., Nappi, M., Riccio, D., and Sabatino, G. (2007). 2d and 3d face recognition: A survey. *Pattern Recognition Letters*, 28(14):1885–1906.

[Ahonen et al., 2006] Ahonen, T., Hadid, A., and Pietikainen, M. (2006). Face description with local binary patterns: Application to face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(12):2037–2041.

[Alexandre, 2012] Alexandre, L. A. (2012). 3d descriptors for object and category recognition: a comparative evaluation. In *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal*.

[Belhumeur et al., 1997] Belhumeur, P. N., Hespanha, J. P., and Kriegman, D. J. (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):711–720.

[Besl and McKay, 1992] Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics.

[Blanz and Vetter, 2003] Blanz, V. and Vetter, T. (2003). Face recognition based on fitting a 3d morphable model. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(9):1063–1074.

[Breiman, 1996] Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

[Bronstein et al., 2003] Bronstein, A. M., Bronstein, M. M., and Kimmel, R. (2003). Expression-invariant 3d face recognition. In *Audio-and Video-Based Biometric Person Authentication*, pages 62–70. Springer.

[Bronstein et al., 2005] Bronstein, A. M., Bronstein, M. M., and Kimmel, R. (2005). Three-dimensional face recognition. *International Journal of Computer Vision*, 64(1):5–30.

[Brown et al., 2005] Brown, G., Wyatt, J., Harris, R., and Yao, X. (2005). Diversity creation methods: a survey and categorisation. *Information Fusion*, 6(1):5–20.

[Bruce and Young, 1986] Bruce, V. and Young, A. (1986). Understanding face recognition. *British journal of psychology*, 77(3):305–327.

[Chang et al., 2005] Chang, K. I., Bowyer, K. W., and Flynn, P. J. (2005). An evaluation of multimodal 2d+ 3d face biometrics. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(4):619–624.

[Chen et al., 2013] Chen, D., Cao, X., Wen, F., and Sun, J. (2013). Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. *Computer Vision and Pattern Recognition (CVPR)*.

[Chen et al., 2006] Chen, W., Er, M. J., and Wu, S. (2006). Illumination compensation and normalization for robust face recognition using discrete cosine transform in logarithm domain. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36(2):458–466.

[Cohen et al., 2003] Cohen, I., Sebe, N., Garg, A., Chen, L. S., and Huang, T. S. (2003). Facial expression recognition from video sequences: temporal and static modeling. *Computer Vision and Image Understanding*, 91(1):160–187.

[Collobert and Bengio, 2004] Collobert, R. and Bengio, S. (2004). Links between perceptrons, MLPs and SVMs. In *Proceedings of the twenty-first international conference on Machine learning*, page 23. ACM.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

[Cover and Hart, 1967] Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27.

[Craw et al., 1987] Craw, I., Ellis, H., and Lishman, J. R. (1987). Automatic extraction of face-features. *Pattern Recognition Letters*, 5(2):183–187.

[Cui et al., 2013] Cui, Z., Li, W., Xu, D., Shan, S., and Chen, X. (2013). Fusing robust face region descriptors via multiple metric learning for face recognition in the wild. *Computer Vision and Pattern Recognition (CVPR)*.

[Davis and Goadrich, 2006] Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM.

[Delac et al., 2005] Delac, K., Grgic, M., and Grgic, S. (2005). Independent comparative study of PCA, ICA, and LDA on the FERET data set. *International Journal of Imaging Systems and Technology*, 15(5):252–260.

[Draper et al., 2003] Draper, B. A., Baek, K., Bartlett, M. S., and Beveridge, J. R. (2003). Recognizing faces with PCA and ICA. *Computer vision and image understanding*, 91(1):115–137.

[Eberhart and Kennedy, 1995] Eberhart, R. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, pages 39–43. IEEE.

[Faltemier et al., 2008a] Faltemier, T. C., Bowyer, K. W., and Flynn, P. J. (2008a). A region ensemble for 3-d face recognition. *Information Forensics and Security, IEEE Transactions on*, 3(1):62–73.

[Faltemier et al., 2008b] Faltemier, T. C., Bowyer, K. W., and Flynn, P. J. (2008b). Rotated profile signatures for robust 3d feature detection. In *Automatic Face & Gesture Recognition, 2008. FG'08. 8th IEEE International Conference on*, pages 1–7. IEEE.

[Fanelli et al., 2011] Fanelli, G., Weise, T., Gall, J., and Van Gool, L. (2011). Real time head pose estimation from consumer depth cameras. In *Pattern Recognition*, pages 101–110. Springer.

[Forman and Scholz, 2010] Forman, G. and Scholz, M. (2010). Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement. *ACM SIGKDD Explorations Newsletter*, 12(1):49–57.

[Gorodnichy, 2005] Gorodnichy, D. O. (2005). Video-based framework for face recognition in video. In *Proceedings of the 2nd Canadian conference on Computer and Robot Vision*, pages 330–338. IEEE Computer Society.

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK.

[Hsu et al., 2003] Hsu, C.-W., Chang, C.-C., and Lin, C.-J. (2003). A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University. Available at: `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide`.

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

[Humphreys et al., 1993] Humphreys, G. W., Donnelly, N., and Riddoch, M. J. (1993). Expression is computed separately from facial identity, and it is computed separately for moving and static faces: Neuropsychological evidence. *Neuropsychologia*, 31(2):173–181.

[Jacobs et al., 1991] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

[Jain et al., 2005] Jain, A., Nandakumar, K., and Ross, A. (2005). Score normalization in multimodal biometric systems. *Pattern recognition*, 38(12):2270–2285.

[Jain et al., 2004] Jain, A. K., Ross, A., and Prabhakar, S. (2004). An introduction to biometric recognition. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(1):4–20.

[Jolliffe, 2005] Jolliffe, I. (2005). *Principal component analysis*. Wiley Online Library.

[Kakadiaris et al., 2007] Kakadiaris, I. A., Passalis, G., Toderici, G., Murtuza, M. N., Lu, Y., Karampatziakis, N., and Theoharis, T. (2007). Three-dimensional face recognition in the presence of facial expressions: An annotated deformable model approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(4):640–649.

[Kazhdan et al., 2006] Kazhdan, M., Bolitho, M., and Hoppe, H. (2006). Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 61–70. Eurographics Association.

[Khoshelham and Elberink, 2012] Khoshelham, K. and Elberink, S. O. (2012). Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454.

[Kirkpatrick et al., 1983] Kirkpatrick, S., Jr., D. G., and Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598):671–680.

[Kittler and Alkoot, 2003] Kittler, J. and Alkoot, F. M. (2003). Sum versus vote fusion in multiple classifier systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(1):110–115.

[Kittler et al., 1998] Kittler, J., Hatef, M., Duin, R. P. W., and Matas, J. (1998). On combining classifiers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(3):226–239.

[Kohonen, 2001] Kohonen, T. (2001). *Self-organizing maps*, volume 30. Springer.

[Lazebnik et al., 2006] Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. IEEE.

[Lee et al., 2003] Lee, K.-C., Ho, J., Yang, M.-H., and Kriegman, D. (2003). Video-based face recognition using probabilistic appearance manifolds. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–313. IEEE.

[Leyvand et al., 2011] Leyvand, T., Meekhof, C., Wei, Y.-C., Sun, J., and Guo, B. (2011). Kinect identity: Technology and experience. *Computer*, 44(4):94–96.

[Li et al., 2013] Li, H., Hua, G., Lin, Z., Brandt, J., and Yang, J. (2013). Probabilistic elastic matching for pose variant face verification. *Computer Vision and Pattern Recognition (CVPR)*.

[Li et al., 2010] Li, Z., Imai, J.-i., and Kaneko, M. (2010). Robust face recognition using block-based bag of words. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 1285–1288. IEEE.

[Lin et al., 2007] Lin, W.-Y., Wong, K.-C., Boston, N., and Hu, Y. H. (2007). 3d face recognition under expression variations using similarity metrics fusion. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 727–730. IEEE.

[Lorensen and Cline, 1987] Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. In *ACM Siggraph Computer Graphics*, volume 21, pages 163–169. ACM.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

[Martinez, 1998] Martinez, A. M. (1998). The AR face database. *CVC Technical Report*, 24.

[Messer et al., 1999] Messer, K., Matas, J., Kittler, J., Luettin, J., and Maitre, G. (1999). Xm2vtsdb: The extended m2vts database. In *Second international conference on audio and video-based biometric person authentication*, volume 964, pages 965–966. Citeseer.

[Metaxas and Kakadiaris, 2002] Metaxas, D. N. and Kakadiaris, I. A. (2002). Elastically adaptive deformable models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(10):1310–1321.

[Mian et al., 2007] Mian, A. S., Bennamoun, M., and Owens, R. (2007). An efficient multimodal 2d-3d hybrid approach to automatic face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(11):1927–1943.

[Moon, 1996] Moon, T. K. (1996). The expectation-maximization algorithm. *Signal processing magazine, IEEE*, 13(6):47–60.

[Nowak et al., 2006] Nowak, E., Jurie, F., and Triggs, B. (2006). Sampling strategies for bag-of-features image classification. In *Computer Vision–ECCV 2006*, pages 490–503. Springer.

[Phillips et al., 2005] Phillips, P. J., Flynn, P. J., Scruggs, T., Bowyer, K. W., Chang, J., Hoffman, K., Marques, J., Min, J., and Worek, W. (2005). Overview of the face recognition grand challenge. In *Computer vision and pattern recognition, 2005. CVPR 2005. IEEE computer society conference on*, volume 1, pages 947–954. IEEE.

[Phillips et al., 2003] Phillips, P. J., Grother, P., Micheals, R., Blackburn, D. M., Tabassi, E., and Bone, M. (2003). Face recognition vendor test 2002. In *Analysis and Modeling of Faces and Gestures, 2003. AMFG 2003. IEEE International Workshop on*, page 44. IEEE.

[Phillips et al., 2000] Phillips, P. J., Moon, H., Rizvi, S. A., and Rauss, P. J. (2000). The FERET evaluation methodology for face-recognition algorithms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(10):1090–1104.

[Pinto et al., 2009] Pinto, N., DiCarlo, J. J., and Cox, D. D. (2009). How far can you get with a modern face recognition test set using only simple features? In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2591–2598. IEEE.

[Provost et al., 1998] Provost, F. J., Fawcett, T., and Kohavi, R. (1998). The case against accuracy estimation for comparing induction algorithms. In *ICML*, volume 98, pages 445–453.

[Queirolo et al., 2010] Queirolo, C. C., Silva, L., Bellon, O. R., and Segundo, M. P. (2010). 3d face recognition using simulated annealing and the surface interpenetration measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(2):206–219.

[Reynolds et al., 2000] Reynolds, D. A., Quatieri, T. F., and Dunn, R. B. (2000). Speaker verification using adapted gaussian mixture models. *Digital signal processing*, 10(1):19–41.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *NATURE*, 323:9.

[Rusu et al., 2008] Rusu, R. B., Marton, Z. C., Blodow, N., and Beetz, M. (2008). Persistent point feature histograms for 3d point clouds. *IAS-10*, page 119.

[Schapire, 1990] Schapire, R. E. (1990). The strength of weak learnability. *Machine learning*, 5(2):197–227.

[Sim et al., 2002] Sim, T., Baker, S., and Bsat, M. (2002). The CMU pose, illumination, and expression (PIE) database. In *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, pages 46–51. IEEE.

[Sinha et al., 2006] Sinha, P., Balas, B., Ostrovsky, Y., and Russell, R. (2006). Face recognition by humans: Nineteen results all computer vision researchers should know about. *Proceedings of the IEEE*, 94(11):1948–1962.

[Sivic and Zisserman, 2003] Sivic, J. and Zisserman, A. (2003). Video google: A text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477. IEEE.

[Snelick et al., 2005] Snelick, R., Uludag, U., Mink, A., Indovina, M., and Jain, A. (2005). Large-scale evaluation of multimodal biometric authentication using state-of-the-art systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(3):450–455.

[Tombari et al., 2010] Tombari, F., Salti, S., and Di Stefano, L. (2010). Unique signatures of histograms for local surface description. In *Computer Vision–ECCV 2010*, pages 356–369. Springer.

[Torres et al., 1999] Torres, L., Reutter, J.-Y., and Lorente, L. (1999). The importance of the color information in face recognition. In *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, volume 3, pages 627–631. IEEE.

[Turk and Pentland, 1991a] Turk, M. and Pentland, A. (1991a). Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86.

[Turk and Pentland, 1991b] Turk, M. A. and Pentland, A. P. (1991b). Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition. Proceedings CVPR, IEEE Computer Society Conference on*, pages 586–591. IEEE.

[van Rijsbergen, 1979] van Rijsbergen, C. (1979). *Information Retrieval*. Butterworth.

[Wohlkinger and Vincze, 2011] Wohlkinger, W. and Vincze, M. (2011). Ensemble of shape functions for 3d object classification. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 2987–2992. IEEE.

[Wolf et al., 2011a] Wolf, L., Hassner, T., and Maoz, I. (2011a). Face recognition in unconstrained videos with matched background similarity. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 529–534. IEEE.

[Wolf et al., 2011b] Wolf, L., Hassner, T., and Taigman, Y. (2011b). Effective unconstrained face recognition by combining multiple descriptors and learned background statistics. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(10):1978–1990.

[Wolpert, 1992] Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, 5(2):241–259.

[Xing et al., 2002] Xing, E. P., Jordan, M. I., Russell, S., and Ng, A. (2002). Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 505–512.

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

[Zhao et al., 2003] Zhao, W., Chellappa, R., Phillips, P. J., and Rosenfeld, A. (2003). Face recognition: A literature survey. *Acm Computing Surveys (CSUR)*, 35(4):399–458.

[Zhou et al., 2003] Zhou, S., Krueger, V., and Chellappa, R. (2003). Probabilistic recognition of human faces from video. *Computer Vision and Image Understanding*, 91(1):214–245.