Graz University of Technology

Institute of Computer Vision and Graphics

Master's Thesis

Person Independent Head Pose Estimation by Non-Linear Regression of Histograms of Oriented Gradients

---

Matthias Straka

Graz, Austria, September 2009

Thesis supervisors

Univ.-Prof. Dipl.-Ing., Dr. techn. Horst Bischof

Dipl.-Ing., Dr. techn. Martin Urschler

Dipl.-Ing. (FH) Markus Storer

Abstract

The estimation of the human head pose in facial images is an important task for a variety of applications such as pose tracking, human computer interaction as well as in facial image analysis. Current developments in this research area are analyzed and two algorithms are selected as candidates for the implementation of a person independent Head Pose Estimation System (HPES).

In addition to that, existing face image databases are evaluated for their usefulness in the training of such a system. The creation of a new database with artificial face images, rendered from 3D laser scans, is proposed for a maximum of flexibility in training.

The first HPES implemented for this thesis consists of a Histogram of Oriented Gradients (HOG) descriptor in combination with a Support Vector Regression (SVR) machine which maps descriptor values to continuous pitch and yaw angles of the head pose. A second algorithm utilizes a biased manifold embedding technique in order to solve the same task in an alternative way. Both methods, as well as a combination of them are thoroughly tested on several publicly available databases.

The results show that the HOG/SVR algorithm outperforms the manifold embedding approach in both head pose estimation accuracy as well as in frontal-pose classification. Furthermore a face detection method is presented which is based on the HOG descriptor.

Keywords. head pose estimation, support vector regression, histogram of oriented gradients, biased manifold embedding

Zusammenfassung

Die Schätzung der menschlichen Kopfpose in Gesichtsbildern ist eine wichtige Aufgabe für Anwendungen wie der Posennachführung, bei der Mensch-Maschine Interaktion sowie bei der Analyse von Gesichtsbildern. Es werden aktuelle Entwicklungen in diesem Forschungsgebiet analysiert und schlussendlich zwei Algorithmen für die Implementierung eines personenunabhängigen Kopfposen Schätzungs Systems (KPSS) ausgewählt.

Bereits existierende Datenbanken mit Kopfbildern werden auf ihre Verwendbarkeit beim Trainieren eines solchen Systems hin untersucht. Um eine maximale Flexibilität beim Training zu erhalten, wird vorgeschlagen eine neue Datenbank aus 3D Laser Scans von Köpfen zu generieren.

Das erste KPSS in dieser Arbeit besteht aus einem Deskriptor mit Histogrammen von orientierten Gradienten (HOG) und einem Support Vektor Regressions (SVR) Algorithmus, der Deskriptorwerte auf kontinuierliche Nick- und Gierwinkel der Kopfpose umsetzt. Eine alternative Methode löst die selbe Aufgabe mittels Biased Manifold Embedding. Beide Methoden, sowie eine Kombination aus diesen werden auf verschiedenen frei-verfügbaren Datenbanken getestet.

Die Ergebnisse zeigen, dass die HOG/SVR Methode dem Manifold Embedding Ansatz sowohl in der Genauigkeit der geschätzen Pose sowie in der Klassifikation der Fronal-Pose überlegen ist. Zusätzlich wird eine Methode gezeigt, wie man einen HOG Deskriptor zum Suchen von Gesichtern in beliebigen Bildern verwenden kann.

Schlagworte. Kopfposen Schätzung, Support Vektor Regression, Histogramm von orientierten Gradienten, Biased Manifold Embedding

Acknowledgments

I want to thank Martin Urschler and Markus Storer for the ongoing support during the many months of writing this thesis. They have given me valuable hints and provided all information that I needed for completing a working prototype of a head pose estimation system.

Many thanks go to Prof. Horst Bischof, who set up the project in cooperation with Siemens PSE in Graz. Thanks to the Biometrics Center for the financial support. Josef Birchbauer, who was the contact person in the company, also contributed his experience and gave valuable hints.

This thesis wouldn't have been possible without the support and understanding of my girlfriend Birgit and all my friends. I also want to thank the people of India who gave me an unforgettable vacational break where I renewed my energy for writing this thesis.

Contents

List of Figures

List of Tables

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Place	Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Ort	Datum	Unterschrift

1 Introduction

1.1 Motivation

The estimation of the 3D pose of a human head from still images is a difficult problem but has numerous potential applications. A Head Pose Estimation System (HPES) can be used e.g. as a pre-processing step for pose independent face recognition [5]. Further applications include head pose tracking in video streams [34], facial image analysis [49] as well as human-computer interaction by using the head orientation as a control input [43]. An example for a head pose estimation is given in Figure 1.1 where an arrow indicates the pose of the head which is specified by the yaw and pitch angle.

Figure 1.1: An image with a head pointing 15° to the right and 20° down

While humans learn to quickly estimate the orientation of a head very early in their life, a computer vision system has to overcome a variety of problems which have challenged researchers and scientists for decades [35]. An ideal HPES should demonstrate robustness to various factors like occlusion, noise, lighting and distortion. Moreover, it should be able to handle the enormous varieties of the human face such as different facial expressions of a single person but also the huge differences of faces of people with different gender, ethnicity and age.

None of these difficulties have stopped the development of a variety of different

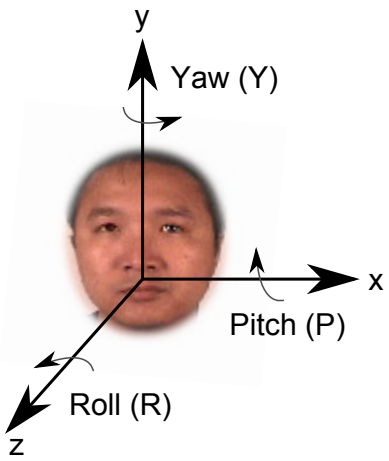


Figure 1.2: Definition of the pose angles

approaches to tackle the challenges of the task. Some algorithms solve the head pose estimation problem with a classification into a fixed number of head poses while other methods aim to estimate a continuous angle in one or multiple degrees of freedom (DOF) [20]. It is obvious that a simple left-right classification requires less complexity than a full reconstruction of the head pose in 3D space. An overview of the different head pose estimation methods can be found in Chapter 2.

1.2 Human Head Pose

A description of the human head pose in a three dimensional space requires at least six DOF. The position of the head in an image is described by its 2D translation and scale. One possible specification of the three rotational DOF of the human head makes use of the following three Euler angles (see Figure 1.2):

- **Yaw angle:** rotation around the vertical (y) axis. Positive yaw angles represent faces where the person looks to his or her left side.
- **Pitch angle:** rotation around the horizontal side-to-side (x) axis. Positive pitch angles represent faces looking up.
- **Roll angle:** rotation around the horizontal back-to-front (z) axis. Positive roll angles represent faces tilted toward the right shoulder.

In Figure 1.3 four typical head poses are shown which correspond to the angles just described. The images were created artificially by rotating a 3D head model in space.

(a) Frontal (b) +45 ° yaw (c) +30 ° pitch (d) -30 ° pitch

Figure 1.3: A face in four different poses

It is important to note that the head pose is not the same as gaze direction. This is illustrated in Figure 1.4 where a face is shown next to its mirrored version. While the left face clearly shows an eye gaze to the right side of the page, the face in the right image looks towards the reader. In fact, the eyes are the same in both images. This illusion is known as the Wollaston Illusion [61]. It is therefore important to distinguish between gaze and pose in an HPES and estimate the pose angle only.

(a) Looking left (b) Looking right

Figure 1.4: Wollaston Illusion [61]: Even though the eyes are exactly the same in both images, the two faces seem to look into two different directions

1.3 Environment for the Head Pose Estimation System

One of the main goals of this thesis is to establish a system that allows frontal pose classification according to the International Civil Aviation Organization (ICAO) standard for machine readable travel documents. Therefore the restrictions and definitions of this standard need to be supported. One of these definitions is the tokenized face image which normalizes all facial images so that the eyes have a fixed position. Only facial images with a frontal pose are accepted in travel documents.

1.3.1 ICAO Specification

The number of international airline travelers is increasing steadily due to the global economy and tourism. In addition, airport security is getting tighter every year. Therefore it is necessary to process identity documents like passports much quicker, more efficiently and possibly automatically while increasing security aspects. A special division of the ICAO for Machine Readable Travel Documents (MRTD) has therefore created an international standard [24] for these documents in order to make them easier to read and process for machines.

The standard describes not only a data format on how to store a photo of a face and its annotations but also restricts which photos are valid in travel documents. These constraints include:

- **Scene constraints** define which requirements the person in the photo must fulfill:
  - A full-face frontal pose with a maximum deviation of $\pm$ 5 degrees from pitch, yaw and roll angle from frontal pose.
  - A neutral (non-smiling) expression with both eyes open and looking into the camera as well as a closed mouth is required.
  - The ICAO standard contains no limitations for the background. It only states that the background should allow a clear separation of the head. It should therefore be uniform with no visible shadows.
  - There should be no significant direction of the light visible in the picture. It is essential that no shadows occlude facial features.
  - Eye glasses are allowed as long as they are transparent and do not contain lighting artifacts like reflections. Also, the frames must not occlude the eyes.

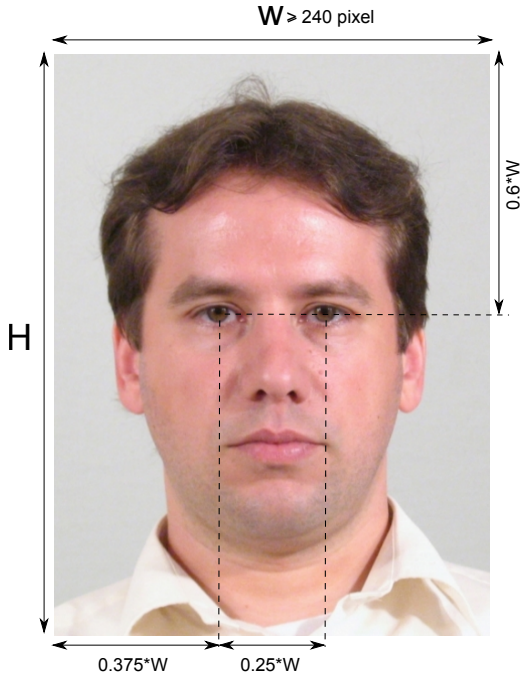


Figure 1.5: The Token Face Image Type format explained on an image

- **Photographic constraints** define the requirements under which the picture must be taken. The image must be correctly exposed so that gradations in the texture are visible in all areas. The subject's eyes, nose, ears and chin must be in focus while a blurred background is allowed.
- **Image constraints** define how the image needs to be recorded and stored. Gray scale images must have at least a 7 bit intensity variation. Color images must allow a gray scale conversion that fulfills the first requirement.

While most of these requirements are valid for frontal poses only, an HPES will be used very early in the verification process in order to classify a given pose as frontal or not. This is why it cannot take most of the requirements for granted. It must work on every face without any assumptions about occlusions, lighting or backgrounds.

1.3.2 Face Tokenization

A very important definition in the ICAO standard is the Token Face Image Type (TFIT). It sets the geometric constraints about where the face in a tokenized image must be located. The only facial features used for this definition are the position of both eyes of the person in the image.

In Figure 1.5 a sample image is shown that fulfills the TFIT format. This tokenization makes processing of the face by a computer very easy because many facial features can be found at almost the same position and scale in any tokenized image. Also, it provides a standardized image for which pose estimation should be performed.

1.3.3 Frontal Pose

According to the ICAO definition, the TFIT must be a frontal face which means that the pitch and yaw angle of the face must be within a 5 degree range of the perfect frontal pose. Unfortunately the ICAO specification lacks one essential definition: the *frontal pose* itself. While there should be no head rotations in this pose, there is no written definition about what zero degree angles mean. Therefore the following convention is used in this thesis (see Figure 1.6):

- The yaw angle is zero, when the face shows a maximum symmetry along the vertical nose-mouth line.
- The pitch angle is zero when the eye corners are on the same height as the upper part of the ear. The easiest way to visualize this is to think of a person wearing glasses. Then the temple of the glasses must be horizontal.
- The roll angle is zero when the connecting line between the two eye centers is horizontal.

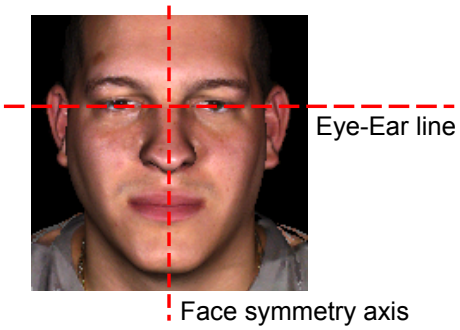


Figure 1.6: Example of a frontal pose image

The definitions presented in this thesis are mostly the same as in the ICAO standard [24] except for the pitch angle which has changed signs (up-angles have a positive pitch value in this thesis).

1.4 Thesis Goals

This thesis investigates some of the most promising approaches to head pose estimation. The development of the HPES is driven by two main questions:

1. What error has to be expected when the yaw and pitch angles of a face are estimated?

2. Can the same system be used to reliably classify faces into frontal and non-frontal poses by thresholding the estimated angles?

While it is possible to design algorithms that are dedicated to each of these goals, this thesis aims to find a unified approach which can handle a continuous head pose angle estimation as well as pose classification simultaneously.

As mentioned in Section 1.3, one application of the HPES will be frontal pose classification according to the ICAO standard. The system assumes that the localization of the face in an image has already been solved and a tokenized image is provided. Pose estimation within -45 and 45 degrees of yaw angle (right/left pose) as well as pitch estimation between -30 and 30 degrees (down/up pose) with an accuracy of around 10 degrees is the goal. While estimating the pose angles in two DOF, the HPES has to fulfill additional requirements such as invariance to different persons, background clutter and robustness towards non-homogeneous lighting.

1.5 Outline and Contributions

The previous sections have defined the environment in which the HPES will be used. This thesis will present the steps that are necessary in order to build such a system. Several types of algorithms and methods are presented in the *Related Work* section of this thesis in Chapter 2. That chapter selects two existing algorithms that are suitable to fulfill the design goals set in Section 1.4.

A complete HPES is more than an algorithm or an implementation of it. It requires a well-balanced set of facial images that can be used for training as well as evaluating the pose estimation performance. There is a wide range of publicly available databases which can be used for this purpose. Chapter 3 gives an overview of these databases. In addition to that, it explains why existing databases are not suited to train an efficient HPES. Therefore a new 2D database derived from 3D laser scans is presented which

allows the training of an HPES that is robust against various sources of disturbances such as random backgrounds and non-uniform lighting.

Chapter 4 starts with an introduction to support vector regression, which is the main machine learning approach used in this thesis. Then two algorithms, which are introduced in the *Related Work* section, are explained in detail. The main part discusses one promising algorithm which uses Localized Gradient Orientation (LGO) histograms [34] and extends it by the Histogram of Oriented Gradients (HOG) descriptor [10] in order to enhance its accuracy for head pose angle estimation. The second algorithm introduces a biased manifold embedding for head pose estimation which may be suitable to enhance the first algorithm even further. At the end of the chapter, an example is given of how the proposed HPES can be extended to not only act as a pose estimation system but also to allow the pose-invariant detection of a face in an image.

A very important part of this thesis can be found in the *Experimental Results* in Chapter 5. There the parameters and settings of the HPES and its algorithms are described and evaluated. A thorough evaluation of the pose estimation performance shows the strengths and weaknesses of the LGO, HOG and manifold embedding techniques. This includes an evaluation on publicly available databases, which demonstrates the real-world behavior of the proposed method.

Finally, in Chapter 6 a summary of the thesis and its results is given. The most important results are summarized in order to emphasize the strengths of the proposed system. In addition to that, an outlook on future work is presented, which gives some hints about possible improvements of the system.

2 Related Work

There exist various different approaches for the determination of the human head pose. In this chapter several categories of algorithms are presented with representative examples. Their main ideas are summarized and advantages and possible disadvantages are discussed. Finally, all approaches are evaluated for their usability in pose estimation in ICAO type images.

According to a survey on head pose estimation by Murphy-Chutorian *et al.* [35] several categories of algorithms exist in the current literature:

Appearance Template Methods: These algorithms work with direct comparison of a new image and a set of exemplar images. The most similar exemplar defines the pose of the new image [3].

The biggest advantage is that it is very easy to add new exemplar images to the pool of existing poses. But this advantage comes with a number of inconveniences. First, only discrete poses can be determined. Also, the larger the exemplar set grows, the more computation time is required to determine the best match. This is why these methods have problems in dealing with the large variability of faces of different persons and emotional expressions. To decrease this problem, filtering the input image with Laplacian-of-Gaussians [16] or Gabor-wavelets [27, 44] is possible.

Detector Array Methods: Algorithms using this method train multiple detectors, each for a discrete pose. The detector with the largest support for a new image defines the pose. This approach is similar to appearance templates with the difference that machine learning is used to model the similarity of images for different poses [22].

The biggest advantage over template methods is that learning algorithms tend to be more robust against appearance variations and put more emphasis on pose related features. Yet it can be difficult to train two classifiers for similar poses. In this case the positive training samples of one classifier must be used as negative samples for the other. Thus they might not learn a useful model of the pose.

This method can be used as a *router* which only estimates the rough pose [42]. It is then succeeded by another classifier that is specific to the detected pose.

Regression Methods: Regression methods can be used to learn a continuous estimation of the head pose by a possibly nonlinear mapping of the image features to pose angles. The biggest problem is the high dimensionality of the input space. This can be solved by using dimensionality reduction algorithms like Principal Component Analysis (PCA) or their kernel extensions [28]. Other regression algorithms like Support Vector Regression (SVR) are able to handle high dimensional input spaces efficiently as well [34].

The advantage of nonlinear regression tools is that they are fast and work well in scenarios where a detailed facial image is available as well as in low resolution images. One of the main disadvantages is the dependence on correct head localization.

Manifold Embedding Methods: These methods assume that even though an image consists of hundreds of dimensions spanned by the pixels of the image, only a few dimensions define the pose [33]. Thus they map the image to a low-dimensional manifold that is defined by the continuous pose. A good algorithm can do this mapping without the influence of variations in faces.

The biggest problem with manifold learning is that it is unsupervised and thus may learn the wrong features. Several approaches have been developed that overcome this problem by making the learning supervised and they show promising pose estimation results [1].

Flexible Models: In contrast to previous methods, flexible models are non-rigid face models that are fit to the 2D image in order to determine the pose. It is possible to adapt the model to the individual face. There exist multiple types of this approach [9, 50]. For example, graphs of facial features can be deformed until they fit a face [62].

The main advantage is that a precise localization of head features is not required initially since these algorithms are able to adapt to the optimal positions. Yet all required facial features must be visible and detectable.

Geometric Methods: These methods rely more on human perception than on appearance based methods. They include measurements of distances between facial features and deviation from bilateral symmetry [60]. In order to be able to handle facial variability, Expectation Maximization (EM) based algorithms are employed.

The advantage of geometric models is that they are fast and simple. But they require accurate localization of facial features like eye corners which may be occluded by glasses, or mouth corners which may or may not exist depending on facial expressions [55].

Tracking Methods: These methods deal with observation of the pose over time. The advantage is that they can track facial features within a restricted area defined by a smoothness of motion constraint. Tracking methods therefore allow a very high accuracy in pose estimation [36, 37]. A major drawback is that most tracking methods require an initialization of the head position. Also, they are obviously not usable for still images.

Hybrid Methods: These methods combine two or more of the previously mentioned approaches.

Apart from a classification into different methods which an algorithm can use, there are basically three types of inputs that can be used for a head pose estimation system:

- **Monocular images:** Here only a single image is presented to the algorithm which must therefore deal with occluded face parts and cannot use any redundant information. A typical area of application is facial image analysis as defined by the ICAO standard.
- **Video streams:** Tracking pose estimation systems deal with multiple images in a sequential manner. Their basic application is the tracking of faces and their poses in a video stream.
- **Stereo or multiple images:** In these systems there is more than one camera which records the pose of a head. The advantage is that redundant data is available which contains far more information than a single camera can capture. Such setups are often found in conference rooms with multiple cameras.

The main focus of this thesis is the continuous head pose angle estimation in monocular still images. A challenge with these images is that a 3D pose estimation needs to be performed from one 2D image only. As there is no 3D information in these images, the estimation problem is ill-posed. In the following sections some important works related to head pose estimation from monocular images are presented.

2.1 Facial Features based Head Pose Estimation

Vatahska *et al.* [55] present an approach that is able to measure the head pose from monocular images by using pose-dependent features. What seems to be a chicken & egg problem is solved by classifying a rough estimate of poses by a face detector. Then the most distinctive facial features for the detected pose are searched in the face region by using Haar-feature based detectors which were trained in a boosting fashion [57]. The final pose (three continuous angles) is then estimated by three previously trained neural networks (one for each pose).

They claim that they are able to solve this task efficiently as they only use distinctive features of the roughly determined pose (e.g. left eye, nose and mouth for a pose where the right eye is possibly occluded). The pose estimation performance depends heavily on the accuracy of all feature detections though.

Wang *et al.* [59] present a distinct approach that models the perspective projection. They try to estimate the point at infinity by using the lines formed by the outer and inner corners of the eyes and the line of the mouth (assuming that both lines are parallel). The vanishing point can then be calculated as the intersection between those two lines. Using this calculated feature, they are able to determine the pose of the head quite accurately. In addition to that they apply an `EM` algorithm that adapts the parameters of their head model to the individual face with its expression. This is done by first applying a previously learned face model to the facial image and then adapting parameters like length and ratios of facial structures.

The adaptation is an advantage of this approach because it allows the method to adapt to the variety of faces in realistic scenarios. One disadvantage is that the model of a fully calibrated camera is required which will not be available in most head pose estimation scenarios. Also, the algorithm cannot perform accurately if parts of the eyes or mouth are occluded (as is the case in extreme left or right poses).

Choi *et al.* [7] present a different approach that utilizes the `EM` algorithm to fit a 3D face model to 2D feature locations. They assume orthographic projections and iteratively

estimate the parameters of the model. Their 3D model of the face is extremely simple and consists of both eyes, the mouth and chin, which are assumed to be coplanar. The advantage of these features is that they are symmetric. In addition, they use the tip of the nose as a point that lies on the perpendicular plane, which goes through the symmetry axis. To obtain the head pose angles, they solve an optimization problem in conjunction with a statistical feature registration method which is executed iteratively.

This approach is suitable to measure pose angles of up to ± 40 degrees. Another advantage is that angles close to the frontal view cause little errors (around 3 degrees). For larger angles these errors increase though as the system suffers from facial feature localization problems.

2.2 Flexible Models

Active Appearance Models (AAM) have been introduced by Cootes *et al.* [8] who claim that by combining enough faces, a dimensionality reduction in terms of variability can be achieved. When head rotation is considered a variability, then it must be among the primary modes of a PCA reduced feature space.

Gui and Chao [19] have solved the problem of pose estimation by using a 2D AAM of the head that adopts to differences in humans as well as changes found in emotional expressions. They propose a two step algorithm that first tries to adopt the model under the assumption that the face is symmetric. As this is not the case because of self-occlusion and noise in facial features, they use a RANSAC [13] method to accurately estimate the rotation matrix which represents the true head pose. As this estimation can be done by solving linear systems of equations, it is quite efficient.

Sung *et al.* [51] present a view-based appearance model solution that adopts the view-based AAM idea of Cootes *et al.* [9] and combine it with 3D morphable models [5]. They train multiple pose dependent face models that are fit to a facial image after a rough pose has been determined. Sung *et al.* have extended the face fitting step by a PCA algorithm that is able to deal with missing data (as described in [18]). This step is important in order to handle occlusion like beards and eye glasses as well as pose related occlusions.

Other flexible methods try to solve the problem of exact face localization by locating facial features using a so-called Elastic Bunch Graph [62]. To find the facial features, a graph with possible feature locations is placed over an image and deformed in order to find the minimum distance of each node with the features in the image. The pose can

then be determined by finding the best matching bunch graph which corresponds to a certain pose.

While this method promises good results due to the adaption to the face, a large number of discrete poses is needed in order to be precise. This makes graph searching rather slow.

2.3 Manifold Embedding

In [29] a head pose estimation algorithm based on linear discriminant models is presented. Li *et al.* argue that the pose can be modeled as a high dimensional manifold and that subspace methods can be used to query the head pose.

For this reason they approximate the non-linearity of the head pose manifold structure with piece-wise linear discriminating subspaces/metrics. As PCA or Linear Discriminant Analysis (LDA) approaches are only able to reduce data in a linear fashion and non-linear methods are not efficient for large data sets, piece-wise linear methods provide a good trade-off.

Other promising Manifold Embedding (ME) approaches include Isomap feature mapping [40], Locally Linear Embedding (LLE) [41] and Laplacian Eigenmaps (LE) [2]. These methods are basically unsupervised and create a non-linear mapping based on the underlying data itself. This is why an improvement has been developed by Balasubramanian *et al.* [1] which biases the ME in such a way that similar poses lie closer together on the manifold.

The biggest problem with these methods is that there is no direct way to map new samples onto an existing manifold. Therefore an additional machine learning method such as the Gaussian Regression Neural Network (GRNN) [65] needs to be used to learn this mapping. Once this problem is solved, the results of almost all manifold embedding methods promise very low angle estimation errors and a fast performance. Unfortunately, this performance can only be achieved on the database which they have been trained on.

2.4 Regression Methods

Murphy-Churtorian *et al.* [34] introduce the idea of LGO histograms. The basic algorithm consists of a face detection step where three AdaBoost-classifiers try to localize either a right or left profile or a frontal face. They use gray scale images to perform these

detections. For each region that captures the whole face, an LGO histogram similar to Scale Invariant Feature Transform (SIFT) descriptors [31] is calculated in order to allow robust correspondence matches. In contrast to object recognition, they use only one LGO histogram which is further smoothed to make it more general.

Three Support Vector Regressors which take these soft LGO histograms as inputs are then trained for each of the three rotation angles of the face. Thus they perform a non-linear regression from feature vectors to angle values.

A similar regression algorithm can be built by utilizing the position of facial features if they are known in advance [32]. Other features that can be used are for example Gabor-wavelets [27].

The main advantage of these methods is that they are fast and require often only face images with the pose angles as a label for training. A disadvantage is that they rely on a good head localization and suffer from shifts in position or scale. Methods like LGO histograms or convolutional networks [38] try to reduce this source of error.

2.5 Conclusion

A thorough evaluation [35] of existing work in the field of head pose estimation has shown that there is a variety of different algorithms that are able to solve the problem efficiently. Each algorithm has its strengths but also its weaknesses.

The next logical step is to select one or two algorithms that can meet the goals defined in this thesis. The main criteria used to choose a suitable method are the continuous angle estimation performance of at least two DOF in images, the robustness to a number of image distortions and the person independent adaptation capability. Especially the interpersonal variance of the human face makes the detection of multiple facial features difficult and therefore error prone. Thus algorithms that do not rely on the accurate detection of these features are preferred. Many algorithms have restrictions such as that they require multiple camera views, video streams or that they only work on a very limited set of persons.

This is why the LGO histogram based approach by Murphy-Chutorian *et al.* [34] is chosen as a promising candidate for an HPES that fulfills the desired goals. Their system does not require any known facial features and relies only on a good localization of the head. As stated in Section 1.4, there already exists an accurate face localization tool which assures this localization. Besides that, a continuous angle estimation in two DOF is possible.

In addition to this descriptor based approach another algorithm is chosen which also looks promising in terms of angle estimation error rates. Especially manifold embedding methods promise a Mean Absolute Error (MAE) of less than five degrees. They also rely merely on a good localization of the head and are therefore comparable to the LGO based method.

Balasubramanian *et al.* [1] have been able to achieve very low estimation error rates on a public database using biased manifold embeddings. Even though they only propose an algorithm that works with one DOF, the extension to multiple DOF is very simple.

In summary both the LGO histogram algorithm and the Biased Manifold Embedding (BME) approach make little assumptions about the underlying image. In fact, they do not even require a face model. This is why this thesis will focus on these two methods and give details about their underlying ideas as well as a thorough evaluation of their pose estimation performance on multiple datasets.

3 Head Pose Databases

This chapter is dedicated to a study of existing face databases. First, the requirements for databases that allow head pose estimation are collected. Then publicly available databases are evaluated with respect to their applicability for this task. Finally, the creation of a custom database is described that fulfills all of the requirements.

3.1 Requirements

For the training of an HPES, a database needs to consist of many images that fulfill certain requirements. The following list gives an overview of the most important requirements that need to be met in order to be useful for head pose estimation:

- A large number of subjects is needed to achieve person independent training. This means that a wide variety of age, ethnicity and gender should be present.
- A fine grained resolution of both pitch and yaw angle is beneficial.
- When several subjects show the same pose (e.g. frontal pose), the pitch and yaw angles should be the same for all subjects. This means that the actual pose should be measured and not be left to the subjective decision of humans.
- The background should be non-uniform so that pose estimation can be trained to perform robustly on images with an arbitrary background.
- In the face images different types of occlusions (eye glasses, facial hair, etc.) need to be present so that the HPES can learn their effects.

- In order to eliminate the face detection task in the HPES, a prior face normalization is desirable which aligns all faces in the database images to the same position, independent of the pose.
- Finally, the database should consist of real human faces in order to train a system that is applicable to real-world scenarios.

3.2 Existing Databases

The list of requirements from the last section shows that building a head pose database requires a careful design. Fortunately, there exist some databases that at least partially fulfill the requirements listed above. Murphy-Chutorian *et al.* [35] present a short overview of existing collections of face images, which provided a good starting point for the evaluation of databases in this section.

3.2.1 Pointing '04

The *Pointing '04* database [17] is a publicly available dataset of different face poses. It consists of 15 different sets of faces where each set contains two series of 93 images. It is claimed that these images contain discrete head poses from -90° to +90° in both pitch and yaw angles. The yaw angles have are resolution of 15 degrees, pitch angles are spaced at 30 degrees.

People of both genders and with different skin colors have been photographed. Also, facial hair and eye glasses are included in some images. Unfortunately no real ground truth is available as the subjects were told to point their head at targets in the room. This leads to a poor uniformity across subjects which introduces a rather large pose estimation error. Also, pitch angles of ± 90 degrees are physiologically impossible and none of the subjects was able to completely reproduce that position. The database is therefore better suited for gaze direction estimation than for head pose estimation.

3.2.2 FacePix 30

The FacePix database [4, 30] has been created by the Center for Cognitive Ubiquitous Computing (CUbiC) of the Arizona State University. It contains 181 images of 30 persons with a yaw-angle span from -90° to 90° in 1° increments. These fine pose angle steps were captured using a motorized camera that turned around the subject's head.

Unfortunately, the database consists only of yaw angle variations, so training for pitch angles is not possible with this dataset alone. All faces are registered so that the eyes and mouth of each face are on the same vertical positions. There is no fixed horizontal position of the eyes across poses though.

In addition to pose images, the database contains lighting variations on frontal face images. For this set of the images a single light source was moved around the head thus generating different light angles in 1 ° increments.

The database is publicly available for non-commercial and educational purposes. At the time of writing this thesis, CUbiC is working on an even larger database containing 1000 individuals.

3.2.3 CMU PIE

The Carnegie Mellon University (CMU) Pose Illumination Expression (PIE) Database was created by Terence Sim *et al.* [45] in the year 2000. They photographed 68 people under 13 different poses and 43 different illumination conditions with 4 different expressions.

As multiple cameras were used to capture the face images, no deviation of the pose between subjects needs to be expected. The database consists of 9 different yaw angles with a 0 ° pitch angle which range from -90 ° to 90 ° at approximately 22.5 ° increments. There are four additional images per setup which have a non-zero pitch angle.

The people in this database were photographed in front of laboratory equipment which acts as background clutter.

3.2.4 FERET

The Face Recognition Technology (FERET) database [39] was created in order to evaluate face recognition algorithms. It contains faces of several hundred people in different poses with varying yaw angle. Each image contains annotations with information about facial expression, presence of eye glasses and hair. The usage is restricted to educational and research work only.

3.2.5 YALE Face Database

The YALE pose and illumination database [15] consists of 10 individuals seen under 64 illumination conditions and 9 poses. For frontal poses, a full annotation of eye and mouth centers is provided. Lighting is performed with multiple spot lights at different

positions around the head. Only one light is activated for each illumination condition. As no fill-light is used to brighten the shadow regions, shadows appear almost black when extreme light angles are used. The YALE database is publicly available for educational purposes.

3.2.6 Siemens Dataset

The Siemens Biometrics Center in Graz has created its own database for ICAO specification related tasks. This database contains images of multiple people in five poses (frontal, left, right, down and up). No exact angle measurements were performed but a simple annotation in terms of discrete pose labels is available.

This database is not publicly available as it is property of the Siemens Austria AG. For evaluation purposes a part of the database was provided to us.

3.3 Creation of a New Database

After a thorough evaluation of different databases it can be said that none of these databases fulfills all requirements for training an HPES. Especially a database that has a fine grained resolution of pose angles in two DOF is not publicly available.

Nevertheless, most of the databases presented in the last section provide a good possibility to evaluate the performance of an HPES once it is trained. Even without exact pose angles or annotations it is possible to evaluate the frontal-pose classification.

To overcome this problem, the creation of a new face pose database is proposed for training an HPES. As the creation of a database with real human faces is very time consuming and expensive, we propose a different method that works with rendered 3D face models from the Binghamton University 3D Facial Expression (BU-3DFE) database [64]. This database consists of 100 subjects, each with a full facial 3D scan including a realistic texture. Each subject was scanned while showing 25 facial expressions. The database has a wide variety of age and ethnicity as well as an equal distribution between both genders. This makes this database ideal for a head pose estimation task.

The creation of a 2D face database from existing 3D data has many advantages:

- No new subjects, laboratory space or camera equipment is needed.
- 3D-Models can be rotated in space, thus any pose can be created. This allows to render as many poses as needed.

- It is possible to simulate different lighting situations.
- Arbitrary backgrounds can be inserted into a rendered image.

The reminder of this section will show the steps that are needed in order to render the images for the training of an HPES.

3.3.1 Virtual Reality Modeling Language

All heads in the BU-3DFE database are stored in Virtual Reality Markup Language (VRML) files, a standardized format for 3D world modeling [23]. These files use plain text encoding and are therefore platform independent. Because of this simple format, VRML files are easy to modify and to extend.

A VRML file (or world) consists of multiple *nodes* which define certain properties of a model. These include 3D vertices, patches, texture mappings, model positions as well as camera and lighting setups. Every node can have a name and sub-nodes thus allowing a tree-like representation of the whole file.

In Listing 3.1 a shortened VRML file from the BU-3DFE database is shown. It defines a head model with the appropriate texture as well as a camera position. Note that every object (like Viewport) has a name which can be used to find and alter its properties (e.g. orientation) from inside an interpreter like Matlab® .

3.3.2 Extending the Database

A VRML file has the advantage of modifiable and extensible nodes. These properties make it easy to adapt the existing files to the requirements of the training database.

Unfortunately the original database lacks a VRML light source and uses only diffuse lighting. Another problem is that the source files do not follow the VRML standard completely and therefore Matlab® refuses to render the head model correctly. There are three important modifications of the original VRML files which need to be performed:

1. Add a directional light which is later used to simulate different lighting situations (see Listing 3.2). The light source is given a name Light which can then be used to access it from Matlab® in order to modify its angle.
2. Convert the texture file (which is originally stored in the .bmp format) to the .png format. This step is needed as the VRML renderer in Matlab® cannot handle non-standard .bmp files. The reference to the file has to be changed accordingly (see Listing 3.3).

Listing 3.1: Example VRML file for a 3D laser scan of a head

```
#VRML V2.0 utf8
DEF Tricorder_object Transform {
    children [
        Shape {
            appearance Appearance {
                texture ImageTexture {
                    url "F0001_NE00WH_F3D.bmp"
                    repeatS FALSE
                    repeatT FALSE
                }
            }
            geometry IndexedFaceSet {
                coord Coordinate {
                    point [54.2842 -66.7874 3.7839, ... ]
                }
                coordIndex [2 0 1 -1, ... ]
                texCoord TextureCoordinate {
                    point [ 0.268409 0.478681, ... ]
                }
                texCoordIndex [0 1 2 -1, ... ]
            }
        }
    ]
}
DEF Tricorder_Camera_Front Viewpoint {
    position 14.3887 -54.6822 202.219
    orientation 0 0 0 3.14159
}
```

3. As the original head shape does not define a `material`-node, a simple material needs to be attached to the `appearance` node of the shape so that the light source can interact with the head model (see Listing 3.3). The ambient light intensity node defines the brightness of shadow regions.

3.3.3 Alignment of the Heads

Unfortunately the 3D models in the `BU-3DFE` database do not have a consistent alignment in 3D space. This is why it is necessary to use an alignment step prior to rendering the heads. Figure 3.1 shows a Matlab® tool that was developed for this thesis which allows

Listing 3.2: Added light node for lighting the head shape

```
DEF Light DirectionalLight {
 ambientIntensity 0.5
 color 1 1 1
 direction 1 0 0
}
```

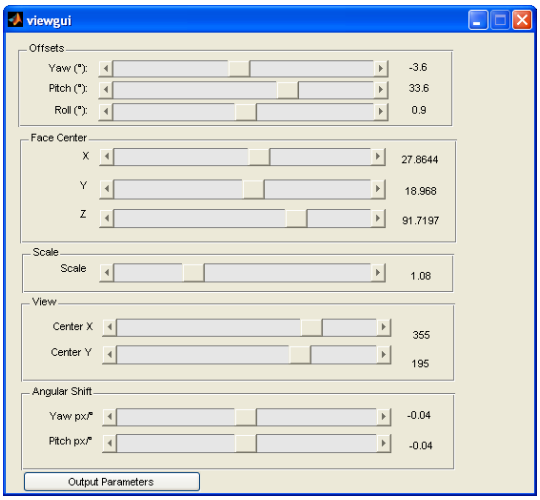
Listing 3.3: Appearance node for the head shape with inserted material node and modified texture format

```
appearance Appearance {
  material Material {
     diffuseColor 1 1 1
     ambientIntensity 0.5
  }
  texture ImageTexture {
     url "F0001_NE00WH_F3D.png"
     repeatS FALSE
     repeatT FALSE
  }
}
```

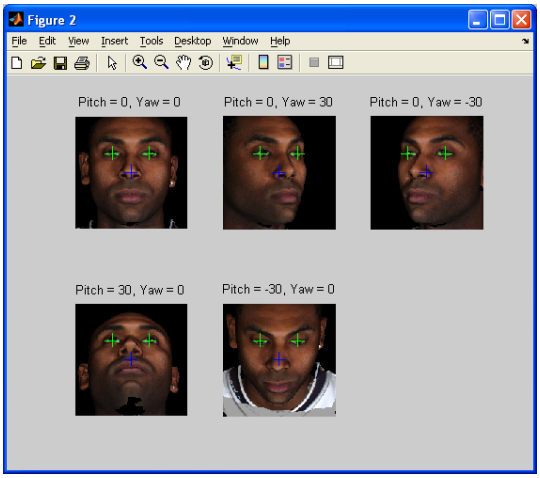
to align all head models so that their eyes have a constant position at all poses within the desired angles.

In Figure 3.1a the control window is shown that allows to set the alignment values by changing the slider positions. There are multiple settings that control the position of the head in space:

- *Yaw, Pitch and Roll offsets*: these values need to be set so that the head in the frontal pose has the correct angles.
- *Face center*: these values are calculated automatically from the `BU-3DFE` annotation files and describe the 3D position of the mid-point between both eyes. All rotations use this point as their center.
- *Scale*: this value defines the size of the head in the image. The correct scale is reached when the distance of the eye centers in the rendered image corresponds to the markers in the frontal pose view.



(a) Control window which allows to align the eyes



(b) Preview of the eye positions at different poses

Figure 3.1: ViewGUI – a tool to align all faces so that their eyes have a constant position across poses

- *View*: these values are a simple shift of the face window in order to center the eyes in the frontal pose.
- *Angular Shift*: these shifts are needed for some heads where eye positions tend to move as the pitch or yaw angle changes. They are therefore an angle dependent correction of the values set in *View*.

These settings need to be tuned for every head model that is used in the final dataset. The process of aligning the faces starts with setting the view position and angular offsets of the face. Then the scale and shifts must be tuned until the eye positions match the cross-marks in all preview images (see Figure 3.1b). As the model files do not change, this tuning needs to be done only once when the database is set up.

3.3.4 Rendering

The database with 2500 individual 3D scans contains far too much variance than is needed. The main focus of this thesis is a person independet head pose estimation. Therefore an equal subset of models m of men and women with only neutral facial expressions is selected to render pose images.

Also, the angles of interest for pose estimation lie between $\pm 45^\circ$ for yaw angles and $\pm 30^\circ$ for pitch angles. This is why there is no need to render angles outside these limits.

As stated in Section 3.1, an HPES with a low angular error requires a grid of poses with about 5° to 10° increment. In order to reduce the similarity between neighboring poses, an equally spaced grid of both pitch ϕ_p and yaw ϕ_y angles is created with 10° increments:

$$I_m(\phi_p, \phi_y) = render(m, \phi_p, \phi_y) \qquad \begin{matrix} \phi_p = \text{-}30^\circ, \text{-}20^\circ, \ldots, 30^\circ \\ \phi_y = \text{-}50^\circ, \text{-}40^\circ, \ldots, 50^\circ \end{matrix} \tag{3.1}$$

In addition to pose parameters, a light-angle parameter ϕ_l is introduced that allows to control the directional light source to rotate around the yaw axis of the head. This parameter is used to create different lighting situations for a head which can be used to make the HPES more robust against the influence of non-uniform lighting. For the rendered poses three light angles were chosen (0° is frontal lighting):

$$I_m(\phi_p, \phi_y, \phi_l) = render(m, \phi_p, \phi_y, \phi_l) \qquad \phi_l = [\text{-}35^\circ, 0^\circ, 35^\circ] \tag{3.2}$$

Using these parameters, each model m is rendered with 7 different pitch angles, 11 yaw angles and three lighting angles which results in 231 images per person. Each image contains color information and has a resolution of 161×161 pixels. Note that none of these images contains any background except black pixels. This makes it very easy to insert various backgrounds later which further increases the number of images per person.

In Figure 3.2 six different faces are shown at different poses and lighting conditions. The complete database which is used for training the HPES in this thesis consists of 39 people at 231 render settings each, which results in 9009 different training pictures.

3.4 Summary

In this chapter the requirements for a face database that is suitable for training an HPES were presented. Some of the most important publicly available databases were studied with respect to their applicability for the training task. Since none of these databases was able to meet the requirements sufficiently, the creation of a new database was proposed.

This new database uses existing 3D scans of human heads with a realistic texture. The 3D models were rendered in multiple poses and a dynamic light source was used

to simulate lighting variations. More than 9000 different images were created using the described method. An alignment step ensures that all rendered faces have the same size and position in the images. This allows the HPES to be trained without a dependence on automatic head localization algorithms.

(a) $\phi_y = 30\,^\circ$, ϕ_p = -20 °, $\phi_l = 0\,^\circ$

(b) $\phi_y = 0\,^\circ$, $\phi_p = 0\,^\circ$, ϕ_l = -35 °

(c) ϕ_y = -30 °, $\phi_p = 10\,^\circ$, $\phi_l = 35\,^\circ$

(d) $\phi_y = 30\,^\circ$, $\phi_p = 30\,^\circ$, $\phi_l = 0\,^\circ$

(e) $\phi_y = 20\,^\circ$, $\phi_p = 10\,^\circ$, $\phi_l = 0\,^\circ$

(f) $\phi_y = 0\,^\circ$, ϕ_p = -30 °, ϕ_l = -35 °

Figure 3.2: Examples of generated head poses at different angle and light settings

4 Head Pose Estimation

In this chapter a detailed presentation of head pose estimation techniques is given. After an introduction to Support Vector Regression (SVR) learning and the description of some important kernels, the main algorithms used for this work are presented. As this thesis is primarily based on the work by Murphy-Chutorian *et al.* [34], their algorithm is studied thoroughly. Then the modifications that lead to HOG based head pose estimation are presented. Also, an alternative method based on Manifold Embedding (ME) is shown. At the end of this chapter, a pose invariant face detection method using HOG descriptors is described.

4.1 Support Vector Regression Learning

Support Vector Regression is one of the most widely used machine learning concepts for the approximation of an arbitrary mapping. This section gives a short introduction to SVR learning and explains the main techniques.

4.1.1 Linear Regression with Support Vectors

Support Vector Regression [11] is a non-linear learning algorithm that is capable of mapping an input vector $\mathbf{x} = [x_1, x_2, \ldots, x_d]^T$ to a scalar output even when the dimensionality d is very high. The basic idea stems from linear regression, which tries to learn the mapping

$$f(\mathbf{x}) = y \tag{4.1}$$

by a linear approximation

$$\tilde{f}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b \tag{4.2}$$

which uses the sum of a dot-product $\langle \mathbf{w}, \mathbf{x} \rangle$ and a biasing term $b \in \mathbb{R}$. This mapping is learned from a training set $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_\ell, y_\ell)\} \in (\mathbb{R}^d \times \mathbb{R})$ with ℓ elements. The training goal is to minimize the estimation error $|y - \tilde{f}(\mathbf{x})|$ while keeping the weights $\mathbf{w} = [w_1, w_2, \ldots, w_d]^T$ as small as possible.

In ε-SVR learning this definition is relaxed so that $\tilde{f}(\mathbf{x})$ can have a maximum deviation of $\varepsilon \in \mathbb{R}$ from its actual value. This leads to the minimization problem

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{subject to} && \begin{cases} y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b \leq \varepsilon & i = 1 \ldots \ell \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i \leq \varepsilon \end{cases} \end{aligned} \tag{4.3}$$

where only errors larger than ε contribute to the estimation error measure.

In order to make SVR more robust against outliers, slack variables $\xi_i, \xi_i^* \in \mathbb{R}^+$ are introduced to allow some data points to stay outside the so-called ε-tube. This leads to a soft-margin loss function

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{\ell} (\xi_i + \xi_i^*) \\ &\text{subject to} && \begin{cases} y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b & \leq \varepsilon + \xi_i \qquad i = 1 \ldots \ell \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i & \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq 0 \end{cases} \end{aligned} \tag{4.4}$$

where $C > 0$ determines the trade-off between the flatness of $\mathbf{w}$ and the amount of errors larger than ε that are tolerated.

The optimization problem can be solved by constructing a *dual problem* with Lagrange multipliers and the use of *quadratic programming*. After some reformulations and intermediate steps (see [47] for more details), a very interesting result is obtained:

$$\mathbf{w} = \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) \mathbf{x}_i \qquad \Rightarrow \qquad \tilde{f}(\mathbf{x}) = \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) \langle \mathbf{x}_i, \mathbf{x} \rangle + b \tag{4.5}$$

where α_i and α_i^* are Lagrange multipliers from the dual problem.

Equation (4.5) is the so-called *Support Vector expansion* which allows to represent $\mathbf{w}$ by a linear combination of ℓ training vectors $\mathbf{x}_i$ (or Support Vectors). This result has two very important consequences: The complexity of the function's representation is independent of the dimensionality of the input space and depends only on the number of Support Vectors. For example, the computational complexity of calculating dot-products scales only linearly with the number of input dimensions d. Many other machine learning algorithms show exponential growth which is known as the *curse of dimensionality*.

In addition to that it can be shown that not all training samples are needed in order to calculate $\mathbf{w}$ thus leading to a sparse representation of Support Vectors.

4.1.2 Kernels

Another important property of Equation (4.5) is that input and support vectors only occur in dot-products. Thus an easy way to perform non-linear regression with an `SVR` is to apply a non-linear mapping $\phi(.)$ to the training patterns $\mathbf{x}_i$.

This may lead to an even higher dimensionality of the input space though. But as seen in Equation (4.5), the `SVR` algorithm depends only on dot-products of the support vectors $\mathbf{x_i}$ and a vector $\mathbf{x}$. It is therefore sufficient to know the kernel

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle \tag{4.6}$$

and not the mapping $\phi(.)$ itself. Now Equation (4.5) can be written as

$$\mathbf{w} = \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) \phi(\mathbf{x}_i) \qquad \Rightarrow \qquad \tilde{f}(\mathbf{x}) = \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) \mathcal{K}(\mathbf{x}_i, \mathbf{x}) + b. \tag{4.7}$$

Two of the most widely used kernels are the linear kernel and radial-basis functions. The linear kernel does not use any mapping and is the same as in Section 4.1.1. It can be written as a simple dot-product:

$$\mathcal{K}_{LIN}(\mathbf{a}, \mathbf{b}) = \langle \mathbf{a}, \mathbf{b} \rangle . \tag{4.8}$$

Another important kernel is based on the Radial Basis Function (`RBF`) with a spread parameter γ which can be used for tuning

$$\mathcal{K}_{RBF}(\mathbf{a}, \mathbf{b}) = \exp\left(-\gamma \|\mathbf{a} - \mathbf{b}\|^2\right). \tag{4.9}$$

The RBF kernel has many advantages as it does not produce any numerical difficulties (values do not rise to infinity), it can simulate the linear kernel for some choices of C and γ [26] and has only one parameter γ that needs to be tuned. A smaller value of γ allows more support vectors to have an influence on a test-vector $\mathbf{x}$, therefore giving a smoother approximation of the unknown function $f(\mathbf{x})$.

4.1.3 Extensions to Support Vector Regression

The main disadvantage of the ε-SVR is the need to choose an adequate value for ε. In [47] a method how to integrate ε into the optimization process is shown. This results in the so-called ν-SVR which automatically adapts the parameter ε to the data. This methods introduces a new parameter ν which can be seen as a weighting factor between errors and the number of support vectors.

4.1.4 Training a Support Vector Regression Machine

SVR machines have the advantage that they are easy to train. Only two steps are required in order to obtain an optimal regression performance [21]:

Dimension wise scaling: In high-dimensional input vectors different numeric ranges for each dimension can occur. This can be a problem in SVR learning because it uses kernels that are calculated as dot-products. During these calculations larger values will dominate over smaller ones.

Therefore linear scaling of each dimension of the input data is advised. The two most widely used pre-processing techniques are:

- shift and scale the data to $[-1, +1]$ or $[0, 1]$ by using the minimum and maximum values of each dimension
- shift and scale the values of each dimension so that it has zero mean and unit variance

The minimum, maximum or statistical values are determined by analyzing the values of each dimension of all vectors from the training dataset. For this thesis, statistical scaling using the mean and variance is used as it is more robust to outliers than minimum/maximum scaling.

Model selection: A ν-SVR with an RBF kernel has three parameters that need to be tuned:

- The cost factor C, which controls the flatness of the weighting vector $\mathbf{w}$
- The kernel parameter γ, which controls the spread of the exponential function
- The adaption control factor ν, which acts as an upper bound on the fraction of errors and a lower bound on the fraction of SVs [47]

In order to find the optimal parameters, a grid-search and cross-validation is recommended [21]. It is practical to increase values for C, γ and ν in an exponential manner (e.g. $C = 2^{-1}, 2^{0}, \ldots, 2^{+3}$).

4.1.5 Software Implementations

There exist several implementations of support vector machines with regression functionalities that have a Matlab® interface. For this work, the open-source package `libsvm` [6] is used because it provides both classification as well as regression with multiple kernels. Another advantage is that the package includes a cross-validation module which can be used for model selection. Compared to other machine learning packages, the runtime performance of the `libsvm` software was found to be fastest while maintaining the same regression performance.

4.1.6 Summary

Support Vector Machines provide a very efficient way of handling high-dimensional input data. They have certain control mechanisms that allow them to be tuned towards optimal performance and robustness against outliers. This makes them the best choice for handling feature vectors like those of descriptors used in head pose estimation.

There exist alternative methods like the Relevance Vector Machine (`RVM`) [53] which is similar to the `SVR` method but uses a Bayesian formulation. The `RVM` training algorithm uses an `EM`-like learning approach which may end up in a local minimum during the optimization process, unlike `SVR` training which is guaranteed to find the global optimum. Also, the computational complexity is very high for a large number of training examples, which is a problem at the training of an `HPES`.

4.2 Localized Gradient Orientation based Pose Estimation

Murphy-Chutorian *et al.* [34] have introduced a Head Pose Estimation System (`HPES`) that is based on localized gradient orientation (`LGO`) histograms and support vector re-

gression (SVR). Their system is the basis for the HPES described in this thesis. The system was chosen because it considers the whole facial region globally and therefore does not require the exact localization of facial features. It only requires a good face localization and promises identity and lighting invariant pose estimation with an accuracy of around 10 ° in both pitch and yaw angles.

In contrast, many pose estimation algorithms require a precise localization of the facial structure, which can be a very difficult task under varying pose, subject and lighting conditions.

4.2.1 Description of the Original Algorithm

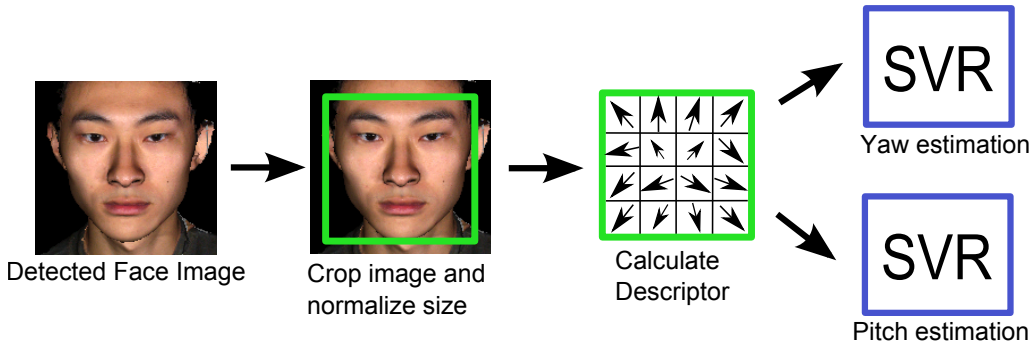


Figure 4.1: Algorithmic blocks of localized gradient orientation based pose estimation

The algorithm as described by Murphy-Chutorian *et al.* [34] basically consists of three steps (see Figure 4.1):

1. First, the location of the face needs to be found in an image.
2. Then, the facial image is resized to a fixed width and height in order to eliminate the effect of scale. For this normalized facial window, an LGO histogram is calculated.
3. Finally, two SVR machines are used to estimate the pose. They are trained to estimate the yaw and pitch angle from the LGO coefficients.

These steps are described in more detail in the following sections.

4.2.1.1 Face Detection

In the original paper Murphy-Chutorian *et al.* [34] propose the usage of three cascaded Adaboost face detectors [57]. There is one detector for a frontal pose which accepts faces

within a $\pm 30^\circ$ pitch angle range. This detector is assisted by two detectors for a left and right side pose. When combined, a range of -80° to 80° in yaw angles can be covered. They state that their detector can detect up to 87% of the faces correctly. While this may be efficient in a video stream, it means that more than one out of ten faces in still images will not be identified correctly.

After face detection, the facial patch is scaled to a fixed size of 34×34 pixels in order to eliminate the effect of scale changes.

4.2.1.2 Feature Calculation

In order to achieve a robust description of the face patch, a soft histogram of localized gradient orientations is calculated. This is essentially a Scale Invariant Feature Transform (`SIFT`) descriptor as introduced to computer vision by Lowe [31]. While Lowe uses multiple `SIFT` descriptors to describe feature points in scale-space in order to perform object recognition, an `LGO` descriptor is used to describe a complete image patch.

The `LGO` descriptor is calculated in the same way as a `SIFT` descriptor. First, the image $\mathbf{I}$ is split into $M \times N$ discrete cells and for each cell a histogram of gradient orientations is calculated. Murphy-Chutorian *et al.* propose the use of a 3×3 Sobel kernel for the calculation of the gradients in horizontal (x) and vertical (y) direction:

$$\nabla \mathbf{I} = \left(\frac{\partial \mathbf{I}}{\partial x}, \frac{\partial \mathbf{I}}{\partial y} \right) = \left(\mathbf{I}_x, \mathbf{I}_y \right). \tag{4.10}$$

Then the orientation of these gradients is used to build a histogram with O bins. For each pixel in the (m, n)-th cell, the orientation is quantized to

$$o_{x,y} = \left\lfloor O \cdot \left(\frac{1}{2\pi} atan2 \left(\mathbf{I}_y(x,y), \mathbf{I}_x(x,y) \right) + 0.5 \right) \right\rfloor \tag{4.11}$$

and used to increment the $(m, n, o_{x,y})$-th histogram bin. $atan2$ is a variation of the arcus tangens which is able to output the correct sign of the angle (positive for counter-clockwise angles). In order to minimize quantization effects, the resulting histogram is smoothed by a three-dimensional kernel.

Finally, all bins are reshaped into a single vector and normalized to unit-length. A truncation of values greater than 0.2 is suggest in order to reduce the effects of single large gradients. The vector is normalized to unit-length once more in order to obtain the final descriptor.

Murphy-Chutorian *et al.* suggest the use of 4×4 cells with 8 orientation bins each ($N = 4$, $M = 4$ and $O = 8$) which are also used in standard SIFT descriptors by Lowe [31]. This choice of parameters yields a 128-dimensional vector.

4.2.1.3 Support Vector Regression Learning

Murphy-Chutorian *et al.* use an SVR method to map the high dimensional LGO feature vector to a continuous angle value. As a kernel, they choose a radial-basis function (see Equation 4.9) as it is one of the most flexible kernels.

For yaw and pitch angle estimation they train two SV regressors. Each regressor is trained with the LGO vectors of the detected training face images as an input, labeled with either the corresponding yaw or pitch angle. They learn the relation between LGO descriptors and angles by the ϵ-SVR training method [47].

Their training set consists of ten individual subjects exhibiting different poses between $\pm 80^\circ$ in yaw angles and -30° to 20° in pitch angles. The grid of discrete poses has a resolution of 5° intervals in both dimensions. In order to tune the optimal regression parameters, they train the SVR with nine subjects and perform a leave-one-out cross validation to evaluate the pose estimation performance.

4.2.2 Summary and Conclusion

The HPES by Murphy-Chutorian *et al.* provides a promising method for head pose estimation. The algorithm consists of only a few powerful steps. They use one of the best head localization techniques combined with a very discriminative descriptor and a strong machine learning approach. In their experimental evaluation they show that a mean absolute error of around 5° in pitch angle and 7° in yaw angle estimation can be reached.

This is why this approach is very suitable as a basis for the HPES proposed in this thesis. The modifications that help to enhance the system are described in the following section.

4.3 Histogram of Oriented Gradients based Pose Estimation

In this work the LGO descriptor included in the HPES by Murphy-Chutorian *et al.* [34] is replaced by the Histogram of Oriented Gradients (HOG) descriptor. It is used to describe

the image patch of a face in different poses in combination with a ν-SVR using an RBF kernel to approximate the mapping from the descriptor values to pitch and yaw angles.

The main structure of the proposed HPES is adapted from the system by Murphy-Chutorian *et al.* which was described in the previous section. It uses the same components as seen in Figure 4.1 and similar processing steps are performed. But in addition to these elements, several improvements are made to increase the performance of the system. The most important enhancements are:

- Gray-level preprocessing in order to reduce lighting effects
- Using a larger scale so that more gradient information can be used
- Usage of HOG features and normalization instead of LGO descriptors
- Better machine learning method in order to adapt better to variations in images

4.3.1 Head Localization

As the main focus of this thesis is to evaluate the head pose estimation performance and not a head localization, no localization is integrated into the HPES. The reason for this is that a head localization step would introduce some errors which can degrade the performance of the head pose estimation.

A 87% success rate on face detection as achieved by the method of Murphy-Chutorian *et al.* [34] is by far not sufficient in a still image facial analysis. Therefore an assumption is made which requires that all images with faces are normalized so that the position of the eyes is constant across poses. While the training database (see Section 3.3) is designed to fulfill this requirement, third party databases can be normalized with existing algorithms [49]. Face tokenization as defined in the ICAO standard [24] is perfectly compatible with this assumption as it requires the eyes to have a fixed location in the image.

Nevertheless, a head localization technique is presented in Section 4.5, which is built upon the HOG descriptor. It is possible to include this head localization step into the processing chain, but only after the descriptor has been trained on previously located faces.

(a) Face without preprocessing (b) Gamma normalization with $\gamma = 0.2$

Figure 4.2: Gamma normalization is used to enhance the brightness in shadow regions

4.3.2 Preprocessing

The first step in the processing chain is the preprocessing of the image. This is very important as it can aid the descriptor calculation. As the pose is not yet known in this step, the preprocessing algorithm must not require prior knowledge of the pose. This makes facial shadow removal algorithms like in [46] not applicable to an HPES.

One suitable algorithm which yields excellent performance on facial images is described by Tan and Triggs [52]. The first part of their method uses gamma correction which is a non-linear per-pixel transformation that maps the gray-scale intensity of a pixel I to the value I^γ for $\gamma \in [0, 1]$. It increases the dynamics of darker regions in the image which ensures that the gradient magnitude in these regions is similar to those in well lit areas. In order not to over-amplify noise a small value for γ is proposed. The default setting in Tan and Triggs' paper is $\gamma = 0.2$ which also seems reasonable for this work. The result of such a normalization can be seen in Figure 4.2.

Subsequent steps of filtering and non-linear contrast normalization as presented by Tan and Triggs do not provide any improvements and are therefore not used. A possible reason for this is that the HOG descriptor already has enough normalization power to even exceed what these preprocessing steps try to achieve.

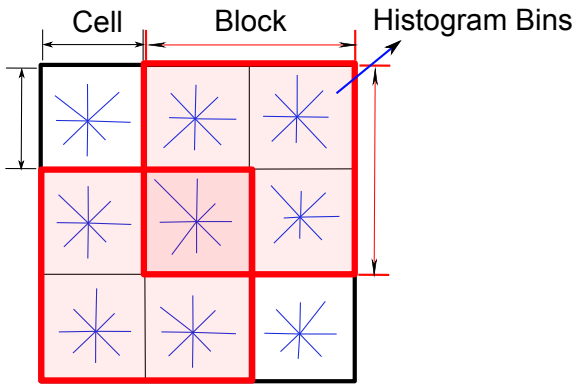


Figure 4.3: Histogram of orientations descriptor elements with $\sigma = 2$ sized blocks and $\beta = 8$ orientation bins

4.3.3 Histogram of Oriented Gradients

The main difference to the work of Murphy-Chutorian *et al.* [34] is the use of an HOG descriptor instead of an LGO histogram. Dalal and Triggs [10] have proven that the HOG descriptor is very effective for human detection. The advantage of the HOG descriptor compared to the LGO descriptor is that not only gradient orientations but also gradient magnitudes are used in the calculation of the descriptor. Also, the HOG descriptor uses an advanced normalization method which is able to reduce the effect of non-uniform lighting.

Another important advantage of the HOG descriptor is the elimination of the smoothing step for the image. While Murphy-Chutorian *et al.* resize the face patch to 34×34 pixels (which is essentially the same as smoothing at a larger scale), Dalal and Triggs suggest using a non-smoothed version of the image at the largest scale possible. They state that smoothing only reduces the influence of strong gradients and that these gradients contain the most discriminative information in an image.

For these reasons, the HOG descriptor can be considered superior in comparison to the LGO descriptor. A more detailed description of the HOG descriptor is given in the following sections.

4.3.3.1 Elements of the Descriptor

An HOG descriptor is a rectangular structure that is applied to a region of an image. It consists of multiple elements that use the pixel information in that region to perform spatial clustering of gradient orientations (see Figure 4.3):

Cells: A cell is square patch covering $\eta \times \eta$ pixels of the underlying image. From these pixels the gradient strengths and orientations are calculated.

Histogram Bins: Each cell contains a gradient orientation histogram with β bins.

Block: A block is a square structure that covers $\sigma \times \sigma$ cells and is used for contrast-normalization. Blocks can overlap each other.

Descriptor: Finally, the complete descriptor consists of all normalized histogram bins from each block. These values are collected in an one dimensional vector.

4.3.3.2 Descriptor Details

The basic image features used are the gray value gradients calculated by filtering with centered derivative operators. As stated by Dalal and Triggs [10], smoothing removes strong gradients and is therefore not desirable. They use a convolution with a simple difference operator to calculate the gradients of an image $\mathbf{I}$:

$$\mathbf{I}_x = \mathbf{I} * [-1, 0, 1] \qquad \mathbf{I}_y = \mathbf{I} * [-1, 0, 1]^T \tag{4.12}$$

The gradient magnitude and orientation ($\arg(.)$) can be obtained for each pixel position by the following calculation:

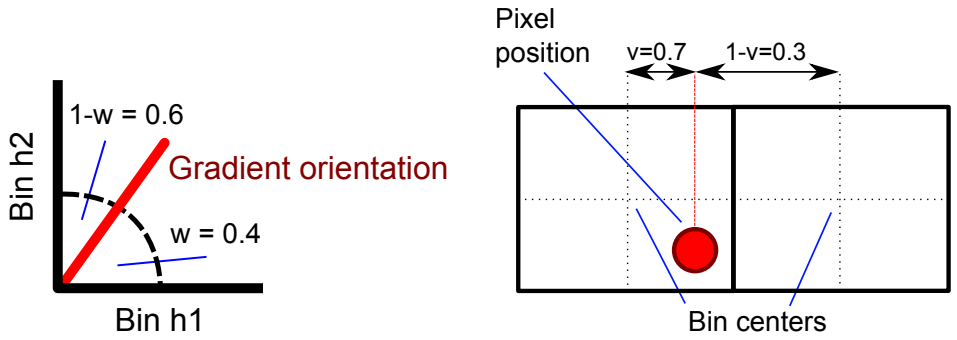
$$|\nabla \mathbf{I}| = \sqrt{{\mathbf{I}_x}^2 + {\mathbf{I}_y}^2} \qquad \arg(\nabla \mathbf{I}) = \arctan\left(\frac{\mathbf{I}_y}{\mathbf{I}_x}\right) \tag{4.13}$$

The next step is to quantize the gradient orientations into β discrete bins. There are two possible ways of doing so:

- signed gradients ($0\,^\circ$ - $360\,^\circ$) use evenly spaced bins on the full circle,
- unsigned gradients ($0\,^\circ$ - $180\,^\circ$) only place bins in the upper half of the circle. Angles above $180\,^\circ$ are mapped to the half-circle bins by subtracting $180\,^\circ$.

In order to avoid quantization effects, angles that lie between two bins (e.g. h_1 and h_2) contribute to both bins using a 1D linear interpolation of the gradient magnitude (see Figure 4.4a).

$$h_1 = h_1 + w \cdot |\nabla \mathbf{I}(x, y)| \qquad h_2 = h_2 + (1 - w) \cdot |\nabla \mathbf{I}(x, y)| \tag{4.14}$$



(a) Orientation contribution of a gradient to angular bins where w is the weight for *Bin 1*

(b) Contribution of a pixel to spatial bins where v is the horizontal weight for the left bin

Figure 4.4: Trilinear smoothing of histogram bins

where $w \in [0, 1]$ is a weight which has the value 1 when the gradient orientation $O_g = \arg(\nabla \mathbf{I}(x, y))$ has the same orientation O_1 as bin h_1. The weight is 0, when the orientation O_g is exactly O_2 of bin h_2. If O_g lies between O_1 and O_2 it is linearly scaled.

$$w = \frac{O_2 - O_g}{O_2 - O_1} \tag{4.15}$$

More details concerning this interpolation are given in [25].

Histogram bins are calculated for every cell of the descriptor. As cells have a discrete rectangular structure, quantization effects need to be treated as well by using a bilinear interpolation between the four nearest cells for every pixel. This is illustrated for the horizontal interpolation in Figure 4.4b where v is a weight similar to w which has the value 1, when the horizontal pixel position lies exactly at the bin center of the left bin. In total, this leads to a trilinear interpolation of the vote of each gradient.

Once all histograms are calculated, block building and normalization complete the descriptor. As demonstrated in Figure 4.3, a block consists of multiple neighboring cells. These cells are normalized as a group after all histogram values of cells in a block b are collected in a vector $\mathbf{v}_b$. Dalal and Triggs present multiple normalization strategies but state that the `SIFT`-style hysteresis normalization proposed by Lowe [31] performs best:

$$\mathbf{v}_b = \mathbf{v}_b / \sqrt{\|\mathbf{v}_b\|^2 + \epsilon^2}. \tag{4.16}$$

Then each value of $\mathbf{v}_b$ is truncated to a maximum value of 0.2. The result is normalized again by using Equation (4.16) where ϵ is a small constant.

The final descriptor consists of all block-normalized histogram bins reshaped into a large vector. Note that some cells appear multiple times in the descriptor but with different normalizations.

4.3.3.3 Implementation Details

For this thesis, a custom implementation of the HOG descriptor has been written in C++ as a Matlab® Executable (MEX) file. There have not been any publicly available MEX implementations that would have had enough flexibility. The MEX file accepts an image, the position of the descriptor as well as the descriptor parameters as input values and returns the normalized descriptor vector.

4.3.4 Support Vector Regression

Similar to the method of Murphy-Chutorian *et al.* [34], an SVR algorithm is used to learn the mapping from descriptor values to pose angles. But in this thesis, a ν-SVR algorithm is used as it allows a better adaptation to the underlying data (see Section 4.1.3).

Dalal and Triggs [10] have evaluated the classification performance of different kernels in their work. An RBF kernel provided the best results but they choose a linear kernel due to runtime considerations in their sliding window approach with a large number of Support Vector Machine (SVM) classifications. As the HPES performs only two SVR estimations there is no need for such a trade-off. Therefore an RBF kernel is chosen as the optimal kernel for the HPES.

4.3.5 Summary and Conclusion

In Figure 4.5 the complete algorithm is shown as a graphical flowchart with all steps starting from the input image to the estimated output angles. The HPES uses a single descriptor in combination with a regression method that is capable to handle high-dimensional feature vectors efficiently.

Compared to the implementation by Murphy-Chutorian *et al.* [34], an even more powerful descriptor is used which handles contrast normalization better. The head localization step is not included in the HPES as it would introduce additional errors in

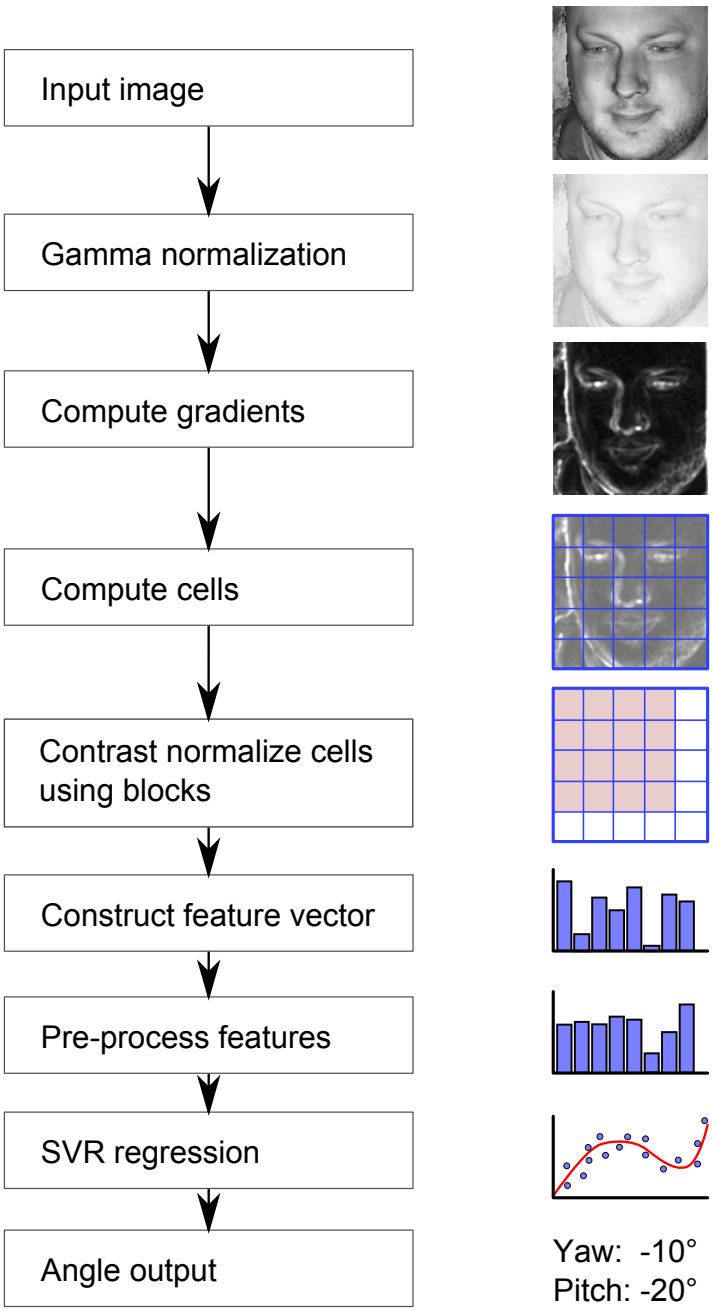


Figure 4.5: HOG/SVR head pose estimation as a flow chart

training. Nevertheless, any algorithm that is suitable to find faces in multiple poses can be used to fill this gap. One such algorithm is presented in Section 4.5.

In the evaluation section (Section 5.2.3) the performance of this system will be evaluated on multiple databases. As a single algorithm will always perform best on one

specific dataset, another HPES algorithm is introduced which will allow some comparisons.

4.4 Manifold Embedding

Manifold Embedding can be used as a different approach for head pose estimation. It is based on the assumption that high dimensional image data can be mapped onto a low-dimensional manifold which models the continuous variation in head pose. New facial images can be embedded into these manifolds. Then, embedded template matching or regression allows to determine the pose of the face [35].

Basically any dimensionality reduction algorithm like Principal Component Analysis (PCA) can be used to perform manifold embedding. But the challenge lies in finding an embedding that correctly maps pose variations while ignoring variations in lighting, occlusions and individuals.

The most promising approaches include Isomap feature mapping [40], Locally Linear Embedding (LLE) [41] and Laplacian Eigenmaps (LE) [2]. They all create an embedding from the training examples without any knowledge about the pose. Therefore the pose must still be extracted from the low-dimensional manifolds by a machine learning approach.

Balasubramanian *et al.* [1] enhanced these manifold embedding techniques by using a *Biased Manifold Embedding* (BME) approach that incorporates pose information into the creation of the manifold. The advantage is that pose differences are weighted much higher than inter-person differences which allows to create a much more robust mapping.

A huge problem of manifold embedding techniques is the mapping of new examples which are not in the training set. Often a direct solution does not exist for this problem. This is why an approximate technique such as a Gaussian Regression Neural Network (GRNN) [65] must be used to learn the mapping from the high-dimensional feature space to the low-dimensional embedding.

4.4.1 Biased Manifold Embedding

Balasubramanian *et al.* [1] have developed a manifold embedding approach based on the Laplacian Eigenmaps (LE) [2] algorithm and extended it by a Biased Manifold Embedding (BME) method. First, the LE embedding is described for arbitrary feature vectors $\mathbf{x}$.

Then the biasing term is introduced. Finally, a method is shown how to perform head pose estimation utilizing such an algorithm.

4.4.1.1 Laplacian Eigenmap Embedding

Laplacian Eigenmaps were introduced by Belkin and Niyogi [2] and are a generic manifold embedding method. Their algorithm is geometrically motivated for representing high dimensional data and allows an efficient non-linear dimensionality reduction while preserving locality properties. This makes them suitable for clustering applications such as a head pose estimation method which tries to keep similar poses within a small neighborhood on a manifold.

Laplacian Eigenmaps can be obtained by a simple three-step algorithm which works as follows:

1. From a set of k d-dimensional vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\} \in \mathbb{R}^d$ a weighted adjacency graph with k nodes is created.

   Two nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ of the graph are connected by an edge if $\mathbf{x}_i$ is among the n nearest neighbors of $\mathbf{x}_j$ (or the other way round). $n \in \mathbb{N}$ is a parameter that can be used to tune the mapping.

   The distance between two nodes is determined by the Euclidean distance

   $$d(i, j) = \|\mathbf{x}_i - \mathbf{x}_j\| \tag{4.17}$$

2. All created edges are weighted by a factor w_{ij}. This can either be an `RBF` function like

   $$w_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{t}} \tag{4.18}$$

   or a simple weight of $w_{ij} = 1$ if two nodes are connected which eliminates the need to tune the spread parameter t (setting $t = \infty$ leads to simple weighting).

3. Finally, the eigenmap is calculated as follows: Compute the eigenvalues λ and eigenvectors $\mathbf{f}$ for the generalized eigenvector problem

   $$L\mathbf{f} = \lambda D\mathbf{f} \tag{4.19}$$

   where D ($D_{ii} = \sum_j W_{ji}$) is a diagonal weight matrix with column sums of W as the entries. $L = D - W$ is the Laplacian matrix.

The solutions to Equation (4.19) are the eigenvectors $\mathbf{f}_0, \ldots, \mathbf{f}_{k-1}$ which are ordered according to their eigenvalues: $0 = \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{k-1}$. As eigenvalue $\lambda_0 = 0$ (because the graph is connected), eigenvector $\mathbf{f}_0$ is discarded and an m-dimensional mapping can be represented as

$$\mathbf{x}_i \rightarrow (\mathbf{f}_1(i), \ldots, \mathbf{f}_m(i)) \qquad m \leq k-1. \tag{4.20}$$

Various implementations of this algorithm exist for Matlab® which can be used for academical purposes*. These implementations are suitable as a basis for the BME enhancement.

4.4.1.2 Biasing the Distance

Balasubramanian *et al.* [1] found out that in classical ME approaches the nearest neighbors of a sample image are more often images of the same person at a different pose than images of different persons at the same pose. As manifold embedding methods are unsupervised, nothing can be done about that.

Therefore they came up with the idea to include information about the pose angles into the distance calculation of Equation (4.17). This additional information is supposed to bias the manifold embedding in such a way that images at the same pose are kept close together while the distance between different poses is artificially enlarged. Also, the ME is created in a way that aids regression tasks like head pose estimation.

Balasubramanian *et al.* include the pose information in a new distance function:

$$\tilde{d}(i,j) = \kappa_1 \cdot d(i,j) + \kappa_2 \cdot f(p(i,j) \cdot g(d(i,j)) \tag{4.21}$$

where $d(i,j)$ is the original Euclidean distance of Equation (4.17) and $p(i,j)$ the pose distance between two images $\mathbf{x}_i$ and $\mathbf{x}_j$. $g(.)$ and $f(.)$ are arbitrary functions and κ_1 and κ_2 are constants. In their work they simply choose $\kappa_1 = 0$ and $\kappa_2 = 1$ as well as $g(.)$ and $f(.)$ to be

$$g(d(i,j)) = d(i,j) \qquad f(p(i,j)) = \frac{|p(i,j)|}{\max_{m,n} p(m,n) - p(i,j)} \tag{4.22}$$

This leads to the BME distance function

$$\tilde{d}(i,j) = \frac{|p(i,j)|}{\max_{m,n} p(m,n) - p(i,j)} \cdot d(i,j). \tag{4.23}$$

*http://www.math.umn.edu/~wittman/mani/, June 2009

Balasubramanian *et al.* only use a one-dimensional pose distance function

$$p(i,j) = |\phi_y(i) - \phi_y(j)| \tag{4.24}$$

which is the absolute difference of two yaw angles $\phi_y(i)$ and $\phi_y(j)$. In a two dimensional HPES with both yaw and pitch angles this function can be easily extended to the Euclidean distance between two (yaw, pitch)-value pairs.

$$p(i,j) = \sqrt{\left[\phi_y(i) - \phi_y(j)\right]^2 + \left[\phi_p(i) - \phi_p(j)\right]^2} \tag{4.25}$$

In principal, this replacement of Equation (4.17) by Equation (4.23) can be performed for any existing manifold embedding approach that works with distances. In this work the Laplacian Eigenmaps [2] method is used as it has proven to be the best method for head pose estimation in a comparison by Balasubramanian *et al.* [1], where they used the biased distance in Locally Linear Embedding [41], Isomap [40] and LE manifold embedding methods.

4.4.1.3 Head Pose Estimation with Biased Manifold Embeddings

In the previous two subsections the essential mathematical framework for manifold embedding was presented. In order to build an HPES, a few more steps are required.

First, the high-dimensional feature vector $\mathbf{x}$ must be defined. Balasubramanian *et al.* use either the raw gray-scale pixel values or the Laplacian of Gaussian (LoG) transformed image. The advantage of the LoG transformation [16] is that it reduces the influence of lighting and identity information because it extracts the edge information from the image.

$$\tilde{\mathbf{I}}(x,y) = |\mathbf{I}(x,y) * LoG(x,y;\sigma)| \tag{4.26}$$

The raw or transformed image must then be reshaped into a single vector for further processing. It is useful to crop a region of the head and resize the image to 32×32 pixels in order to avoid feature vectors that are too large.

The feature vectors of all training images can then be used to create a manifold embedding. In addition to the feature vector, each image must be supplied with yaw and pitch angle information for biasing the distance. The manifold embedding requires as a parameter the number of nearest neighbors n as well as the number dimensions m to which the input space should be reduced.

Once the manifold is created, there is no direct possibility to map an image that is not in the training set onto the manifold. Balasubramanian *et al.* suggest to use a Gaussian Regression Neural Network (GRNN) to learn the mapping from the high-dimensional feature space to the low-dimensional embedded space. This method has been successfully applied to similar problems before [65]. It is basically a network of Radial Basis Functions (RBF) which can be trained to learn the mapping from one space into another without an error on the training set. The only parameter is the spread σ of the GRNN which controls the smoothness of the RBF. For this thesis the GRNN implementation included in the Matlab® library is used.

The feature vector to manifold mapping needs to be learned with the same training images that were used to create the manifold. It is later needed to transform new images so that they lie on the manifold.

The final step in manifold-based head pose estimation is the mapping from the m-dimensional manifold to the pose angles for pitch and yaw. This is why a machine learning algorithm is required that learns this mapping. In order to be comparable to the HOG method, the same SVR algorithm as in Section 4.3.4 is used.

4.4.2 Summary

Manifold embedding looks very promising and achieves very good results for head pose estimation [1]. The main idea is to create a map where each feature point is connected with the closest neighbors. This map is then used to calculate dimensionality-reduced features by solving a generalized eigenvector problem.

An HPES can benefit from this approach as distances between poses and individuals can be biased so that a person independent system which is sensitive to pose changes only can be built. The real world performance of such a system will be evaluated in Section 5.3.

4.5 Face Localization Algorithm

The localization of a face in an arbitrary image is a challenging task considering the multi-pose nature of human heads. Many detectors achieve good performance only on near frontal poses [49]. This is why the HPES proposed in this thesis assumes that the head localization has already been solved in order not to introduce an additional error source.

Various approaches exist that try to solve the detection and pose estimation task simultaneously in order to take advantage of synergy effects between the two problems [28, 38]. Unfortunately these techniques cannot be integrated into the previously presented HOG approach.

Nevertheless, this section presents an algorithm that takes advantage of the existing framework for HOG descriptor based pose estimation. With only some modifications and extensions a reliable face detector can be built by reusing the HOG descriptor as a feature vector for face classification.

4.5.1 Algorithm

The algorithm presented here is similar to the method of Dalal and Triggs [10]. They demonstrate that the HOG descriptor can be used in a human body detection task. In fact, there is not much difference between the detection of a human and a face in an image. This is why the same approach is used as a face detection algorithm.

Basically, a linear SVM classifier is trained to discriminate between HOG descriptors of various facial poses and arbitrary background patches. For this task the same HOG descriptor as in pose estimation can be used with the same set of parameters. Even the size of the descriptor can be left unchanged as the window it describes contains the whole face.

A sliding window extracts multiple HOG descriptors within a region of interest (ROI) (which might be the whole image when no assumptions are made). Then a previously trained SVM is used to classify each window into face or non-face. An SVM is able to not only output a binary decision but a classification score which is positive when a face is detected. It reaches a maximum when the sliding window reaches the exact face position. A non-maximum suppression based on this score therefore allows to select the best position for each detected face and eliminates multiple detections of the same face.

In Section 5.8 a qualitative demonstration of this approach is given. As this thesis focuses on pose estimation rather than face localization, there is no extensive evaluation nor optimization for this approach.

4.5.2 Possible Improvements

As stated in [28], training the classification on all poses simultaneously is not very efficient. Better results can be expected when training several detectors and combining

their results. For example, this could be an SVM for left-poses only in combination with a mirrored version of the search-image for full-pose detection.

Another problem is a poor runtime performance even for small search regions when compared to face detection algorithms such as in [56]. Possible optimizations therefore include the use of integral image representations and cascaded HOG descriptors as in [66] where the computation time for one image is reduced significantly while maintaining almost the same detection rate.

4.5.3 Conclusions

The simple face detection approach presented in this section demonstrates that the HOG descriptor can be used for both regression of pose angles and face classification or pose invariant face detection. Unfortunately, this approach cannot compete with other multi-view face detection algorithms in terms of runtime performance.

4.6 Summary

In this chapter the algorithms used for head pose estimation were studied. A short introduction to Support Vector learning presented the ideas and strengths of kernel based non-linear regression. This machine learning approach can be used to learn the mapping from HOG descriptors to pose angles. In addition to this approach, an alternative approach that tries to solve the same task by means of manifold embedding was presented.

All these methods promise an accurate head pose estimation. The following chapter will therefore verify this promise and thoroughly test the proposed algorithms for their pose estimation ability as well as the performance in a frontal-pose classification task as required by the ICAO specification.

Finally, a face detection algorithm based on HOG descriptors was introduced that demonstrates the wide-spread areas of usage for the gradient based description method.

5 Experimental Results

The following sections evaluate the performance of the algorithms in this thesis and try to show how precise the pose can be estimated at different pitch and yaw angles as well as which classification performance can be expected for the frontal-pose detection. They present the experimental setup, optimize the parameters for the algorithms and show performance evaluations and comparisons of the different approaches.

This chapter does not only evaluate the Histogram of Oriented Gradients (HOG) descriptor based approach but also studies the performance of a manifold embedding approach. There are three variations of the Head Pose Estimation System (HPES) which use different descriptors and learning methods:

1. an HOG descriptor using an SVR for regression (see Section 4.3)
2. a BME approach with LE embedding (see Section 4.4)
3. a hybrid approach which combines manifold embedding and the HOG descriptor

In addition to a thorough evaluation of these systems there is an evaluation of the robustness of the proposed approach. Finally, the HOG based face detection algorithm is evaluated on several databases.

5.1 Evaluation Setup

All evaluations are performed in a similar setup in order to assure that the results of different algorithms can be compared objectively. In this section a description of the methods and datasets used for the experiments is given.

5.1.1 Training and Test Datasets

In order to train an HPES, training data that represents the wide variety of poses, interpersonal differences, background and lighting situations is required. Therefore most evaluations are performed on the artificially generated head pose image database which is described in Section 3.3.

This database is split into a *training set* which is used to train the HPES, and a *test set* that allows to evaluate the pose estimation performance. While both datasets are from the same database, they contain head images from different individuals and are therefore independent from each other. This selection guarantees that the HPES does not only perform good on individual persons but shows good generalization performance across individuals. If not stated explicitly, all HPES are trained with the *training set* (or a part of it) and the performance results are obtained by testing the HPES on the complete *test set*.

In addition to that, the Siemens database (see Section 3.2.6) is used to evaluate the frontal pose classification by means of Receiver Operating Characteristics (ROC) curves. Note that the Siemens database does not contain any explicit pose angles but is rather classified into up, left, down, right and frontal pose. While frontal pose images have perceived yaw and pitch angles closely to zero degree, non-frontal poses show a wide variety of angle values.

5.1.2 Evaluation Measures

In order to compare the performance of different head pose estimation systems, a method of measurement is needed. The aim is to obtain significant measures for both the head pose estimation as well as the classification scenario.

5.1.2.1 Pose Estimation Performance

A typical measure for the accuracy of an HPES are the mean absolute error (MAE) values E_p and E_y of the estimated pitch and yaw angles as well as the standard deviation σ_p

and σ_y from these mean values [35].

$$\Delta\phi_y(i) = \left|\Phi_y(i) - P_y(\mathbf{D}_i)\right| \qquad \forall i = 1 \ldots N \tag{5.1}$$

$$E_y = \frac{1}{N}\sum_{i=1}^{N}\Delta\phi_y(i) \tag{5.2}$$

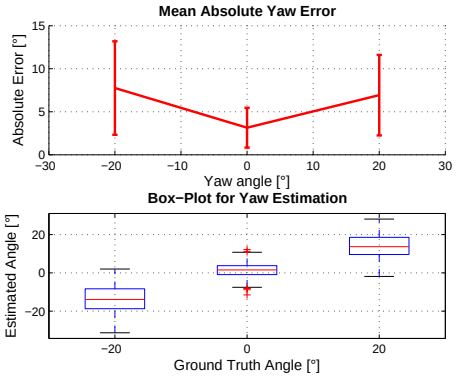
$$\sigma_y = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\Delta\phi_y(i) - E_y\right)^2} \tag{5.3}$$

where $\Phi_y(i)$ is the true yaw angle and $P_y(\mathbf{D}_i)$ the predicted yaw angle from a descriptor $\mathbf{D}_i$. The same calculations need to be performed for the pitch angles. In order to obtain the statistical error values, the pose angles need to be estimated for all N pose images from the *test set*. Single statistical measures, such as mean and standard deviation, often hide weaknesses of the system, therefore two types of graphs are generated (see Figure 5.1a):

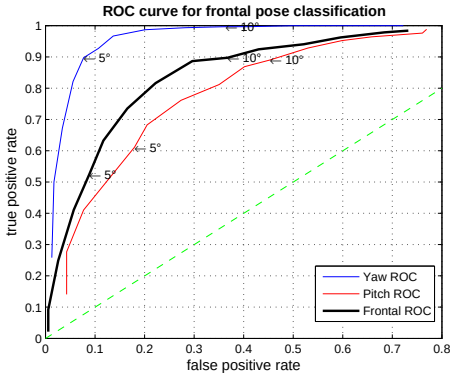
- The *mean absolute error plot* shows the mean absolute error in combination with its standard deviation for both the yaw and pitch angle. As a complete evaluation in both yaw and pitch space would require a three dimensional grid-like structure, a grouping method is used: For yaw error plots multiple discrete yaw angles φ_y are chosen and plotted along the x-axis. The value on the y-axis is the mean absolute error of all $\Delta\phi_y(i)$ with a yaw angle of φ_y and arbitrary pitch angles φ_p. For example, in Figure 5.1a the values -20 °, 0 ° and 20 ° are chosen for φ_y. The same method is used to evaluate the pitch angle estimation.
- A *boxplot* is added in order to allow an analysis with robust statistics [54]. The boxplot shows the median of estimation errors as well as the lower and upper quartile which build a box around the median value. The same grouping strategy as for the mean absolute error plot is used.

5.1.2.2 Frontal Pose Classification

In addition to the estimation error, ROC plots (as in Figure 5.1b) will be shown that evaluate the frontal-pose classification performance. This classification is performed through a prediction of both the yaw and pitch angle of the head with a final thresholding by a



(a) Yaw estimation error plot



(b) Receiver operating characteristics plot

Figure 5.1: Example plots for the evaluation of head pose estimation and frontal pose classification performance

single threshold value $T_{frontal}$:

$$F_y(i) = \begin{cases} true & |P_y(\mathbf{D}_i)| < T_{frontal} \\ false & \text{otherwise} \end{cases} \qquad F_p(i) = \begin{cases} true & |P_p(\mathbf{D}_i)| < T_{frontal} \\ false & \text{otherwise} \end{cases} \tag{5.4}$$

$F_y(i)$ and $F_p(i)$ are boolean variables that are true when the predicted pose of the i-th image from the *test set* is classified as frontal. For ground truth data $\tilde{F}$, the frontal pose is defined as a pose with an angle less than $\tilde{T}$ on both the pitch and yaw axis:

$$\tilde{F}_y(i) = \begin{cases} true & |\Phi_y(i)| < \tilde{T} \\ false & \text{otherwise} \end{cases} \qquad \tilde{F}_p(i) = \begin{cases} true & |\Phi_p(i)| < \tilde{T} \\ false & \text{otherwise} \end{cases} \tag{5.5}$$

As a ground truth threshold value $\tilde{T}$ a frontal angle of $5\,^\circ$ is chosen according to the ICAO definition [24]. The number of true positives TP and false positives FP is then calculated as the number of correctly classified poses for all N elements from the *test set*:

$$TP = \sum_{i=1}^{N} \left(F_y(i) \wedge \tilde{F}_y(i)\right) \qquad FP = \sum_{i=1}^{N} \left(F_y(i) \wedge \neg\tilde{F}_y(i)\right) \tag{5.6}$$

and

$$TN = \sum_{i=1}^{N} \left(\neg F_y(i) \wedge \neg\tilde{F}_y(i)\right) \qquad FN = \sum_{i=1}^{N} \left(\neg F_y(i) \wedge \tilde{F}_y(i)\right) \tag{5.7}$$

for true negatives TN and false negatives FN respectively.

Having a binary decision (frontal vs. non-frontal), an ROC curve can be plotted that shows the effect of the threshold value $T_{frontal}$ on the true positive rate (TPR) and false positive rate (FPR). These values can be calculated as follows [12]:

$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{FP + TN} \tag{5.8}$$

The value of $T_{frontal}$ is varied from 1° to 15° in one degree increments. In addition to that the 5° and 10° points are marked by an arrow.

While yaw and pitch ROC curves describe the performance for each angle separately, the so called *frontal ROC* uses both the yaw and pitch angle to determine the frontal pose:

$$F_{frontal}(i) = F_y(i) \wedge F_p(i) \tag{5.9}$$

Note that the frontal ROC is not used in all evaluations as it might make the plots hard to read in some cases.

5.1.3 Evaluation Procedure

A complete evaluation of an HPES consists of multiple steps which are summarized in the following list:

- The HPES is trained with all images from the *training set*. If a training algorithm cannot handle this large number of images (which is the case for manifold embedding methods), a random subset of these images can be used for training.
- After training, several evaluations are performed on the *test set* as well as on other databases:
  - For all face images from the *test set*, the yaw and pitch angles $P_y(\mathbf{D}_i)$ and $P_p(\mathbf{D}_i)$ are estimated. Then the mean absolute estimation error E_y and E_p as well as their standard deviation σ_y and σ_p are calculated after the descriptor $\mathbf{D}_i$ has been determined.
  - If the face images do not originate from the *test set* database (e.g. Siemens database), they need to be scale-normalized first. This is done by resizing them so that the eye distance in the images matches the eye distance in the *test set* images (which is 60 pixels).

- After the pose has been estimated for all images, the error plots as in Figure 5.1a can be generated from the error values $\Delta\phi_y(i)$ and $\Delta\phi_p(i)$. This is not possible for databases that do not contain ground-truth angles.
- In addition to that, an ROC plot as in Figure 5.1b is generated which shows the frontal-pose classification performance.

This standard evaluation allows a unified comparison of all methods and settings for the head pose estimation systems investigated in this thesis.

5.2 HOG/SVR Based Head Pose Estimation

The first system that is evaluated is the HPES proposed in Section 4.3 and consists of an HOG descriptor and an SVR learning algorithm to map the descriptor values to yaw and pitch angles. This section will choose the optimal parameters for the system and evaluate its performance on multiple databases. In addition to that, the experiment from the original HPES, which uses an LGO descriptor, is replicated using the *training* and *test set*.

5.2.1 Choice of Parameters

An HOG descriptor requires some parameters that need to be tuned. This section will investigate the influence of various descriptor parameters and find the optimal parameters for head pose estimation. All experiments are performed by using the *training dataset* and the performance is evaluated on the *test set* for both yaw and pitch angle estimation.

5.2.1.1 Descriptor Size

The most important parameters describe the size and position of the HOG descriptor on a facial image. The position is chosen so that the descriptor is centered between the eyes and lies at the height of the nose in a frontal pose image. This choice ensures that the descriptor will cover the full facial region at all possible poses.

In order to determine the optimal size of the descriptor window, an experiment is performed where the width and height of the descriptor region is varied. At a given eye distance of 60 pixels in the images from the *training/test set*, the performance of the descriptor is studied at a width/height of 90, 120 and 150 pixels. The mean absolute pose prediction errors for pitch and yaw angles can be seen in Figure 5.2. The lowest errors

can be observed at a descriptor width of 120 pixels, which is two times the distance of the eyes in the images.

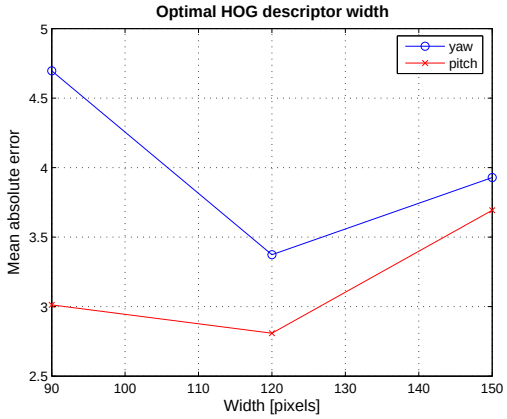


Figure 5.2: Evaluation of HOG descriptor window width at an eye distance of 60 pixels

While this setting captures the face perfectly in a near-frontal pose, ear regions and other facial properties can be missing in profile-views. This behavior is intended as ear regions are very often occluded by hair and therefore do not contain reliable information. Also, the wider the descriptor region is chosen the more background is visible, which can reduce the descriptor performance.

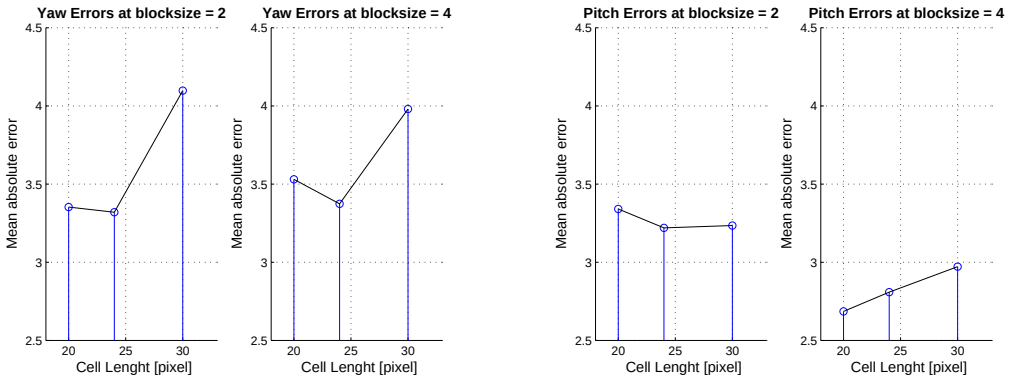
5.2.1.2 Cell and Block Sizes

After the optimal size of the descriptor has been determined, its cell size and block normalization parameters need to be tuned. Therefore a parameter search is performed that varies the horizontal and vertical length of the cells between 20 and 30 pixels for a face descriptor region of 120 pixels. This has a direct impact on the actual number of cells which can be calculated as follows:

$$\#\text{cells} = \frac{\text{descriptor length}}{\text{cell length}} \qquad \text{e.g.} \quad \frac{120\,\text{pixel}}{20\,\text{pixel}} = 6\,\text{cells} \tag{5.10}$$

In addition to different cell sizes, the effect of the block-size was studied at blocks of 2×2 and 4×4 cells.

Figures 5.3a and 5.3b show the mean absolute estimation errors on the *test set* after training with the parameters just described. It can be seen that using square cells with a length of 24 pixels yields the best result in both pitch and yaw estimation. While there is no improvement by reducing the cell width to 20 pixels, a performance decrease can



(a) Cell width for yaw estimation

(b) Cell width for pitch estimation

Figure 5.3: Determination of the optimal cell lengths and block sizes for the HOG descriptor

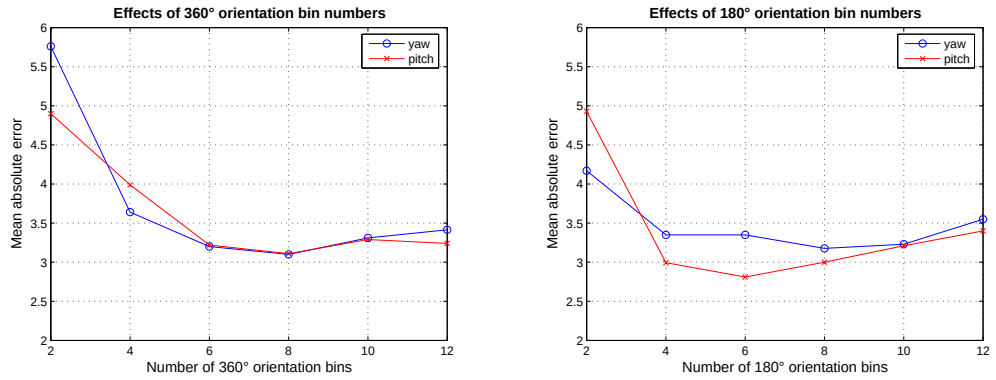
be noted by increasing their size to 30 pixels. Another observation is that using a larger block size reduces the error because of better contrast normalization.

The optimal spatial parameters are therefore a grid of 5×5 suqare cells with a width of 24 pixels which are normalized by 4×4 blocks. Experiments on additional databases support these findings as the `HPES` with these parameter settings shows lower mean absolute errors compared to other settings.

5.2.1.3 Orientation Binning

The last parameter that needs to be determined is the number and type of gradient orientation bins for the histograms. Figure 5.4 therefore shows the performance of two gradient orientation binning methods at increasing bin numbers. In Figure 5.4a the performance of signed 360° degree orientations is plotted. It shows an optimal value at 8 orientation bins. A smaller value causes a significant increase in the error while larger values do not improve the result. When the binning is performed with unsigned 180° orientations (as in Figure 5.4b), almost the same performance can be measured compared to the signed version.

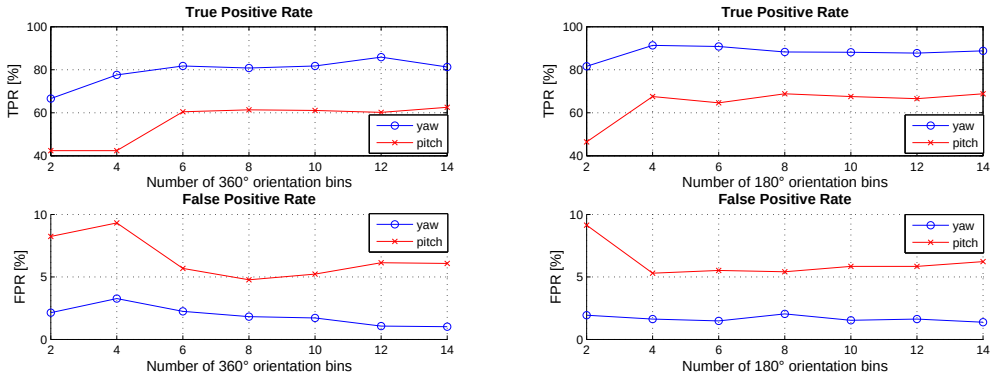
Therefore another experiment was performed, which evaluates the frontal pose classification performance of the `HPES`. In Figure 5.5 the percentage of correctly detected frontal poses as well as the percentage of false positives at a threshold of $T_{frontal} = 5^\circ$ is evaluated for different bin numbers as well as signed and unsigned gradients. It can be



(a) Signed 360 ° gradient bins

(b) Unsigned 180 ° gradient bins

Figure 5.4: Influence of the number of HOG orientation bins on pose estimation



(a) Signed 360 ° gradient bins

(b) Unsigned 180 ° gradient bins

Figure 5.5: Influence of the number of HOG orientation bins on frontal pose classification

seen, that the unsigned gradient method outperforms the HOG with signed gradients in frontal pose classification.

For this reason, the unsigned gradient orientation binning is proposed for head pose estimation. Additional experiments have shown that unsigned gradient orientation binning performs better than the signed version at frontal pose classification. A number between 8 and 10 orientation bins should be used for the descriptor in order to perform in an optimal way. These parameters reflect the values suggested by Dalal and Triggs [10] for their human detection task. Note that a lower number of orientation bins

might as well be sufficient when a small increase in the MAE can be tolerated. However, this has a negative effect on the frontal pose classification.

5.2.1.4 Support Vector Regression

As stated in Section 4.3, a ν-SVR using an RBF kernel is a good choice for head pose estimation. This type of algorithm is chosen because of the self-adaptation of the ν-SVR to noise in the data.

The parameters for this machine learning algorithm need to be determined with a grid search method. The optimal settings can be found in Table 5.1. They are used for training the HPES in all evaluations.

Table 5.1: Optimal parameters for training the ν-SVR for pose estimation with HOG descriptors

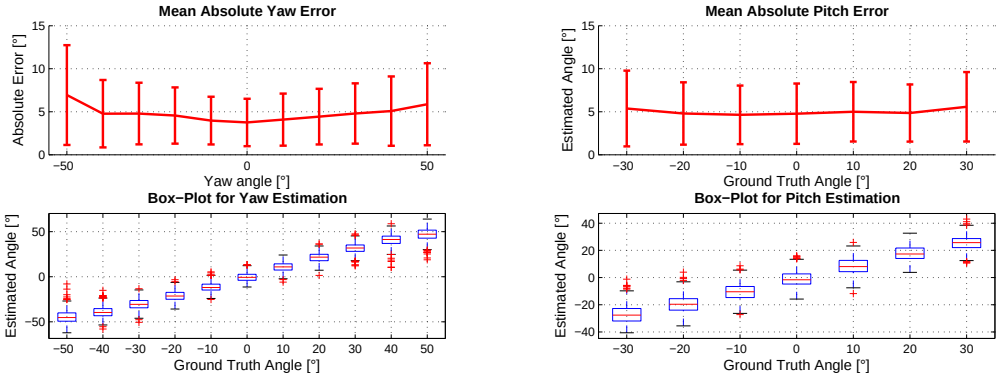
Type	Parameter	Yaw estimation	Pitch estimation
Smoothness	C	$2^4 = 16$	$2^3 = 8$
RBF spread	γ	$2^{-10} = 0.00098$	$2^{-9} = 0.0019$
Adaptation control	ν	$2^{-5} = 0.03$	$2^{-5} = 0.03$

5.2.2 Replication of the LGO Descriptor Results

Prior to evaluating the head pose estimation performance of the HOG descriptor system, the experiment performed by Murphy-Chutorian *et al.* [34] is replicated. Therefore the HOG descriptor is exchanged by an LGO descriptor, parametrized with the same settings used by Murphy-Chutorian *et al.* (see Section 4.2). The descriptor uses 4×4 square cells with a width of 8 pixels and eight signed gradient orientation bins. This descriptor is calculated from a facial patch which is downsized to 34×34 pixels.

In Figures 5.6 and 5.7 the head pose estimation performance as well as the ROC evaluation on the *test set* and Siemens database is shown. As the training data used in the original paper is not publicly available, the *training set* described in this work is used for training the HPES.

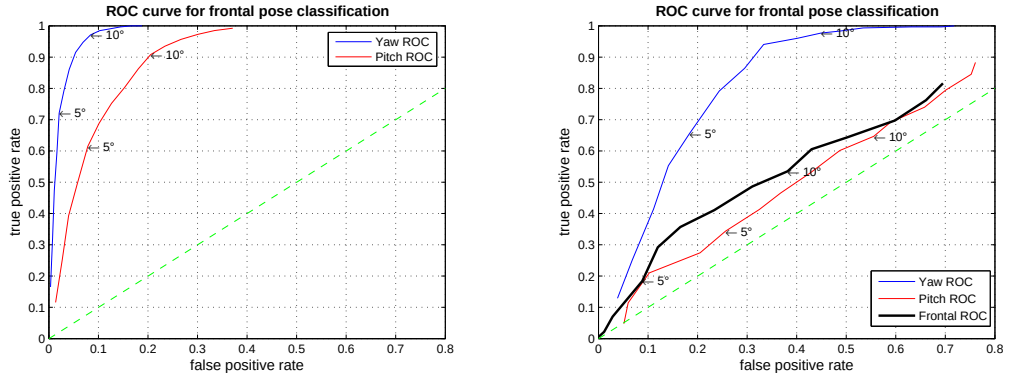
With a mean absolute yaw angle error of $E_y = 4.8°$, the result comes close to the findings of Murphy-Chutorian *et al.* with an MAE of $4.67°$ for yaw angles. For pitch angles this experiment shows a slightly worse performance at $E_p = 5°$ compared to $3.39°$ in the original paper. The reminder of this section will show that using an HOG



(a) Estimation performance for yaw angles ($E_y = 4.8, \sigma_y = 3.8$)

(b) Estimation performance for pitch angles ($E_p = 5.0, \sigma_p = 3.7$)

Figure 5.6: Performance of the LGO descriptor with SVR learning on the test set



(a) ROC on the Testset

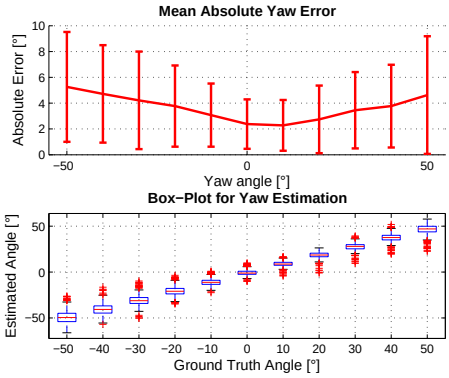
(b) ROC on Siemens dataset

Figure 5.7: Receiver operating characteristics for the LGO descriptor based system

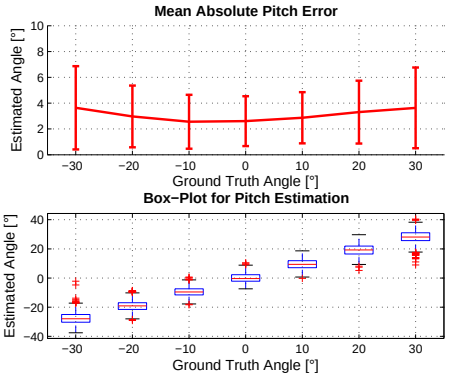
descriptor instead of the `LGO` descriptor improves both the head pose estimation as well as frontal pose classification.

5.2.3 Performance Evaluation of the HOG Descriptor

Figures 5.8 and 5.9 show the results for the evaluation procedure with an `HOG` descriptor of a total width and height of 120 pixels. It uses 5×5 square cells with 24 pixels in width. The unsigned gradient orientations are binned into nine bins for each cell. The

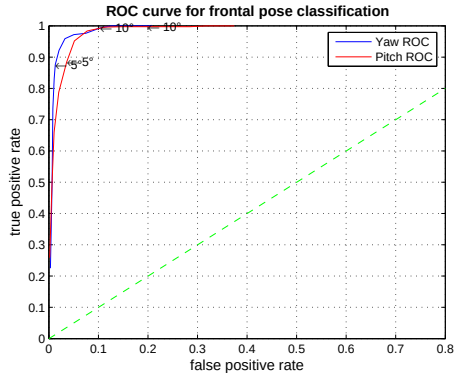


(a) Estimation performance for yaw angles ($E_y = 3.6, \sigma_y = 3.3$)

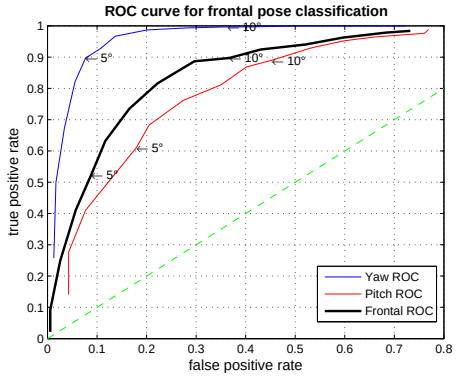


(b) Estimation performance for pitch angles ($E_p = 3.1, \sigma_p = 2.6$)

Figure 5.8: Performance of the HOG descriptor with SVR learning on the test set



(a) ROC on the Testset



(b) ROC on Siemens dataset

Figure 5.9: Receiver operating characteristics for the HOG descriptor based system

descriptor is normalized by four overlapping 4×4 blocks.

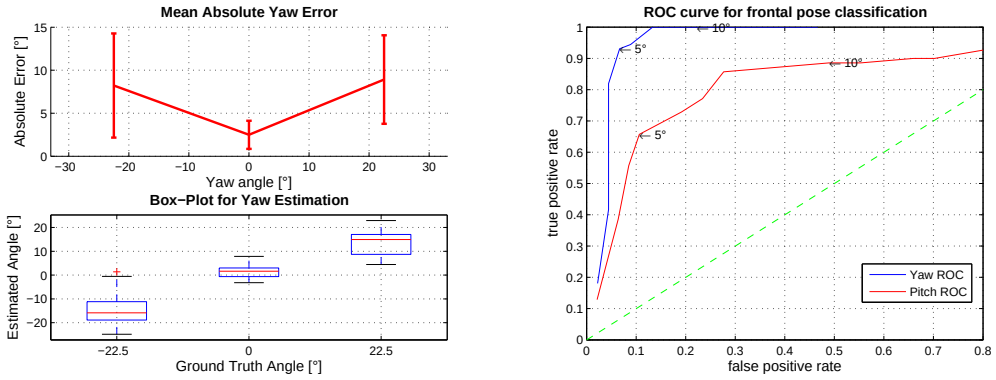
Compared to the results of the LGO descriptor in Section 5.2.2, an decrease of the mean absolute error for both pitch and yaw angles can be seen on the same data. Especially at near frontal poses the estimation error is significantly reduced which leads to a better frontal pose classification in both the *test set* as well as on the Siemens database.

An effect that can be observed for both descriptors is the increase of the MAE as well as the associated standard deviation when the yaw angle increases. This is due to the fact that in a profile pose more background is visible. The *test set* contains random images as

background clutter which will influence the HOG descriptor and reduce its performance. Even though the SVR attempts to learn this variance it cannot be completely eliminated.

5.2.4 Performance on Additional Datasets

The real world behavior of the HPES using an HOG descriptor with ν-SVR learning can be demonstrated on publicly available databases. These databases are described in detail in Chapter 3.



(a) Mean absolute errors for different yaw angles

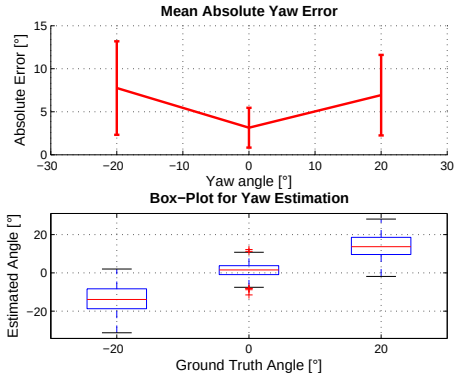
(b) Receiver Operating Characteristics

Figure 5.10: Performance of the HOG descriptor on the CMU PIE database

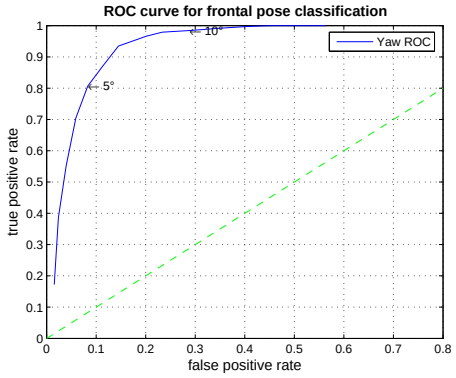
Figure 5.10 shows results on the CMU PIE database [45] . Especially the yaw estimation in frontal poses is very good which explains the low false positive rate on the ROC curve.

All images were tokenized by the algorithm described in [49]. Note that face tokenization only works accurately on frontal poses. Non-frontal faces can therefore have a localization error which affects the performance of pose estimation as well. While the CMU database offers a wider range than $\pm 22\,^{\circ}$, angles larger than this were left out of the evaluation as no usable face localization was possible using the tokenization method. The face localization from Section 4.5 was not used because it introduces some deviations due to incorrect localization. This would also have artificially decreased the frontal face detection rate of the system.

In Figure 5.11 the same type of evaluation has been performed on the FERET dataset [39]. The face images were also tokenized before pose estimation. Again, the ROC curve shows a good frontal face detection at a low false positive rate. In the FERET



(a) Mean absolute errors for different yaw angles



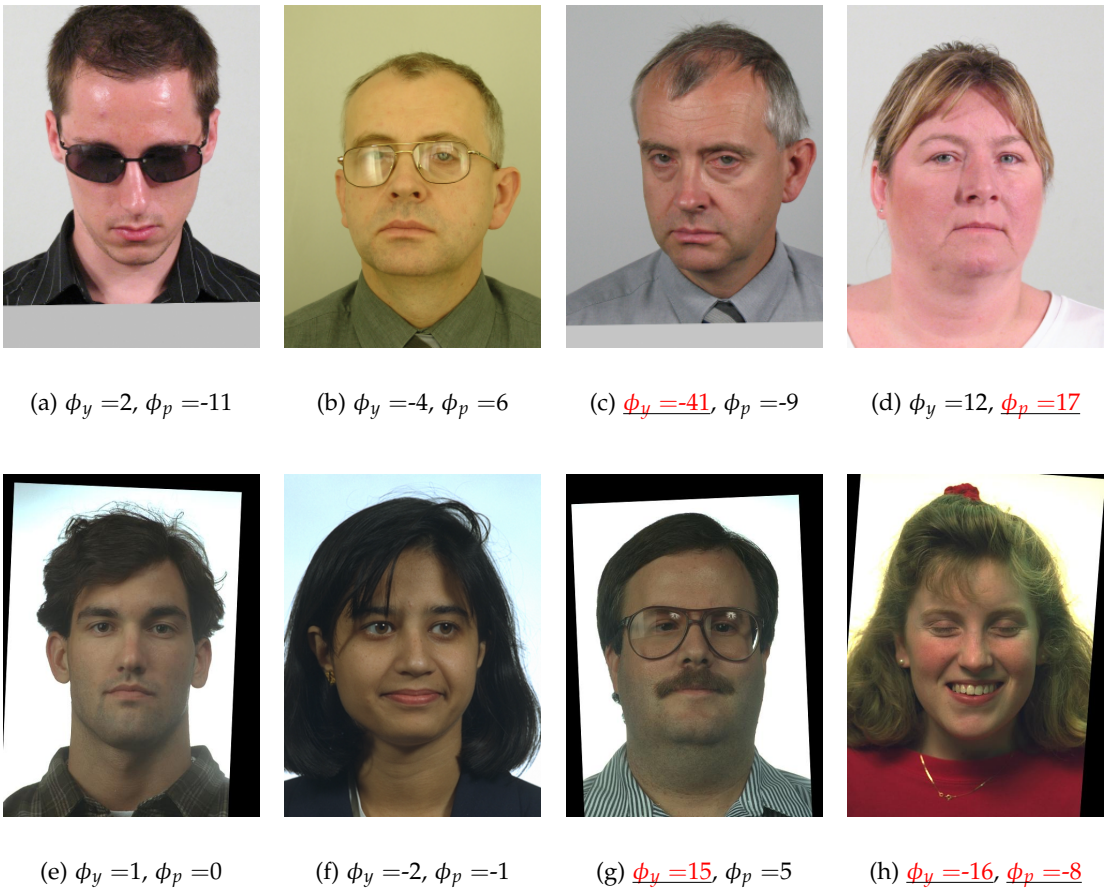
(b) Receiver Operating Characteristics

Figure 5.11: Performance of the HOG descriptor on the FERET database

database there is no pitch-variation, so only the evaluation of the yaw estimation was possible.

In Figure 5.12 several example pose estimations for the Siemens dataset (images a to d) and FERET dataset (images e to h) are shown. An estimated angle with an absolute error of larger than 10° is shown in red letters and with an underline. It can be seen that facial hair and eye glasses often lead to an incorrect estimation. Face images with such occlusions are not included in the training database due to the lack of appropriate 3D models in the BU-3DFE database. Including such images into the training set can therefore reduce the pose estimation error on the example images and all evaluations. Nevertheless, a correct pose angle estimation is possible on some face images with glasses as shown in Figures 5.12a and 5.12b.

Finally, Figure 5.13 shows the evaluation on the FacePix 30 database [4, 30]. The frontal-pose detection yields almost the same results as on the previously presented datasets. As the FacePix dataset already includes a face localization, no position correction was performed. Unfortunately, the face positions are not correctly centered at larger yaw angles. This property of the database, which is described in Section 3.2.2, causes the MAE to rise in non-frontal poses. A simple way to overcome this problem is presented in Section 5.8.2.



(a) ϕ_y =2, ϕ_p =-11 (b) ϕ_y =-4, ϕ_p =6 (c) ϕ_y =-41, ϕ_p =-9 (d) ϕ_y =12, ϕ_p =17

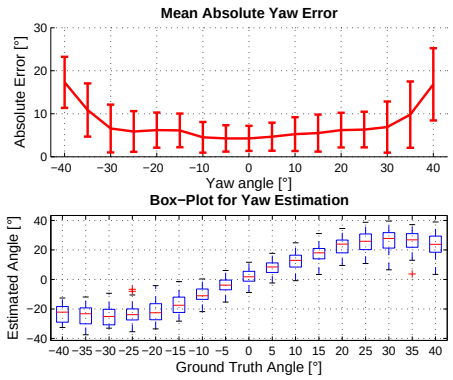
(e) ϕ_y =1, ϕ_p =0 (f) ϕ_y =-2, ϕ_p =-1 (g) ϕ_y =15, ϕ_p =5 (h) ϕ_y =-16, ϕ_p =-8

Figure 5.12: Failure and success on single face images

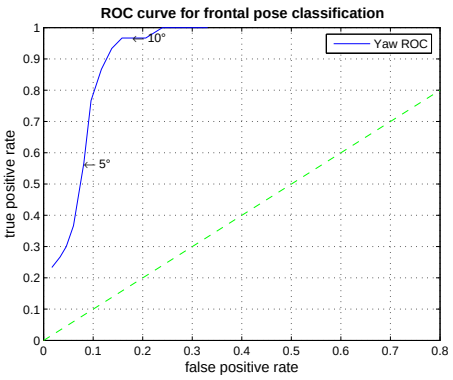
5.2.5 Conclusion

The HOG descriptor with an SVR learning algorithm shows a good head pose estimation performance on the *test set* as well as a reliable frontal-pose classification. The assumption that the HOG descriptor yields a better performance than using an LGO descriptor to extract a feature vector from an image patch has been verified. For the HOG descriptor, mean absolute errors of below 10° are possible for all tested databases (with the exception of the FacePix database due to a face localization problem).

When used for frontal pose classification, the system is able to classify yaw angles on more than 90% of the faces correcly at a false positive rate of 10%. Unfortunately this result could not be achieved for pitch angle classification.



(a) Mean absolute errors for different yaw angles



(b) Receiver Operating Characteristics

Figure 5.13: Performance of the HOG descriptor on the Facepix database

5.3 Manifold Embedding

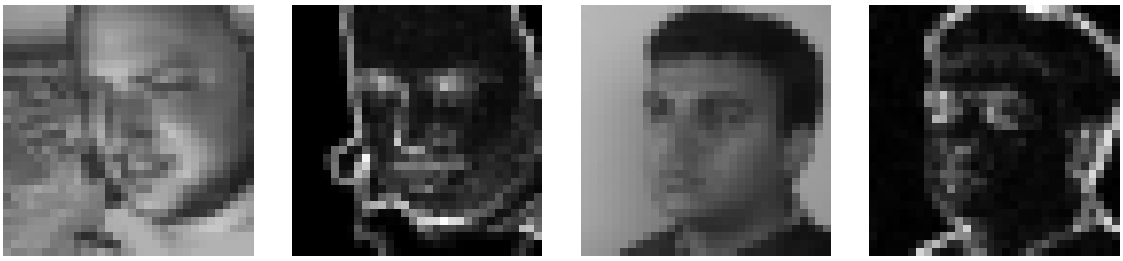
The following section evaluates the pose estimation performance of the Biased Manifold Embedding (`BME`) method presented in Section 4.4. First, the findings of Balasubramanian *et al.* [1] are replicated on the FacePix database. Then, the *training set* images are used to train the `BME` system and its performance is compared to the `HOG/SVR` method.

5.3.1 Choice of Parameters

The most important parameter for manifold embedding is the dimensionality of the manifold. Balasubramanian *et al.* [1] show that there is no significant increase in performance at a dimensionality at $m = 50$ or higher. While a larger number of dimensions will not improve the estimation performance significantly, a much longer runtime needs to be expected due to the `GRNN` mapping. Therefore the dimensionality of $m = 50$ is a good trade off between accuracy and execution speed.

Another important variable in manifold embedding is the size of the *training set*. The final manifold will contain all data points that are used for training. No reduction similar to the sparse support vectors in `SVR` is possible. Therefore only 1000 randomly sampled images from the *training set* are used to build the manifold in order to avoid out-of-memory problems.

The last parameter that can be tuned is the number of nearest neighbors for the creation of the map. Here the value $n = 50$, as suggested by Balasubramanian *et al.*, is



(a) Sample gray image from the *test set* with background

(b) Sample LoG image from the *test set*

(c) Sample gray image from the FacePix dataset

(d) Sample LoG image from the FacePix dataset

Figure 5.14: Example of images that are used as feature vectors in biased manifold embedding

used as it has proven to be a good choice.

5.3.1.1 General Regression Neural Networks

Manifolds do not have a direct method to map new high-dimensional vectors onto an existing low dimensional manifold. This is why `GRNN` learning is utilized to learn this mapping as good as possible [65].

A `GRNN` requires only one parameter, the spread σ. The larger σ is chosen, the smoother the approximation of the mapping will be. Matlab® provides an easy way to create a two-layered `GRNN` with the function `newgrnn`.

The optimal value of σ can be found by splitting the data that needs to be mapped into a training set and a test set and performing a parameter search with mean squared error evaluation on the test set. The best choice is a value of $\sigma = 1$.

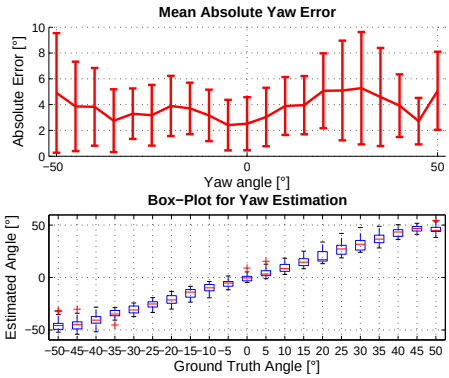
5.3.2 Performance Evaluation

5.3.2.1 Replication of Biased Manifold Embedding Results

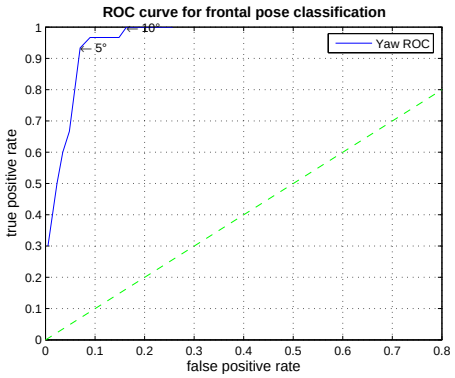
As stated in Section 4.4, the `BME` method by Balasubramanian *et al.* [1] is very promising and the authors show some of the best `MAE` rates up to date on the Facepix database. In order to replicate these findings, the following steps and settings were to used:

- The 128×128 pixel FacePix images were filtered with a 7×7 LoG filter with $\sigma = 1$

- Then they were resized to 32×32 pixels without any previous cropping (see Figure 5.14d). The 1024-dimensional feature vectors consist of the pixel values of these images.
- The images of the first 10 individuals from the FacePix dataset were used to test the HPES while images from the other 20 individuals were used for training
- A Laplacian eigenmap biased manifold was created using $n = 50$ nearest neighbors and a reduction to $m = 50$ dimensions.



(a) Estimation performance for yaw angles ($E_y = 3.8, \sigma_y = 2.9$)



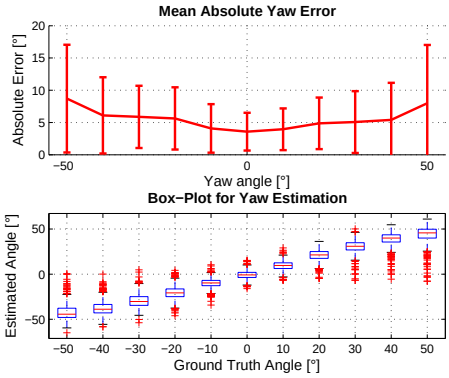
(b) ROC plot for frontal pose classification

Figure 5.15: Replication of the BME results of Balasubramanian *et al.* on the FaxePix dataset

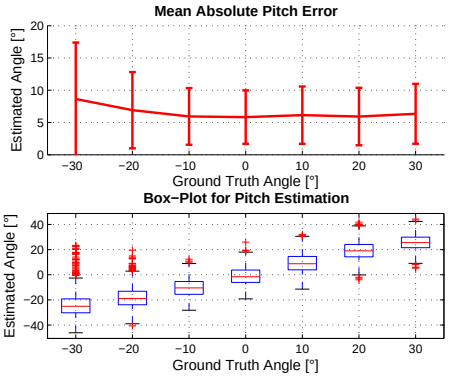
In Figure 5.15 the results of the replication of the experiment on the FacePix database [4, 30] is shown. With an MAE of 3.8 ° the BME method almost outperforms the HOG descriptor implementation. The question is, if these results can be achieved on a different database with an additional DOF, the pitch angle.

5.3.2.2 Performance on the Training/Test Set

The results in Figures 5.16 and 5.17 were obtained by training an HPES with the method of Balasubramanian *et al.* from the *training set* with pose angles in two DOF. In order to be comparable to the HOG/SVR method certain steps needed to be altered. Only then the method is directly applicable to other databases and comparable with HOG/SVR results:

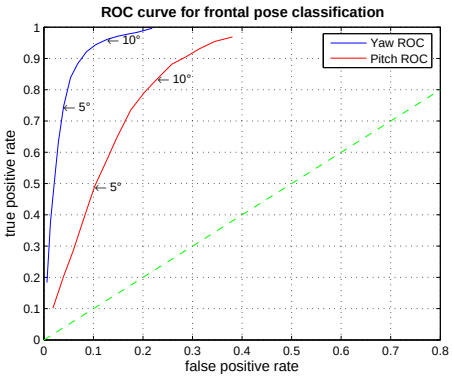


(a) Estimation performance for yaw angles ($E_y = 5.6, \sigma_y = 5.7$)

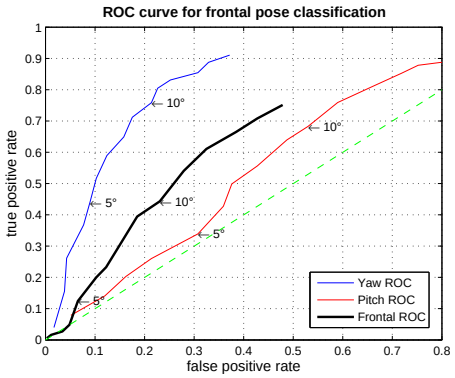


(b) Estimation performance for pitch angles ($E_p = 6.5, \sigma_p = 5.5$)

Figure 5.16: Performance of the Manifold Embedding of LoG images on the Testset



(a) ROC on the test set



(b) ROC on Siemens dataset

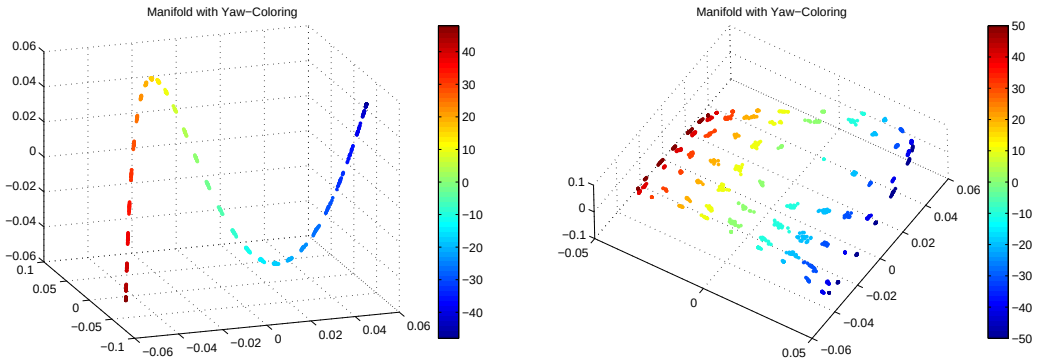
Figure 5.17: Receiver operating characteristics for Manifold Embedding on LoG images

- The same descriptor region as for `HOG` descriptors is used. This square region is centered on the face and has a width of 120 pixels, which is twice the distance between the eyes (see Figure 5.14a).
- A 7×7 LoG filter with $\sigma = 1$ is applied in order to reduce lighting effects.
- The filtered region is then downsized to a 32×32 image.
- From this image a 1024 dimensional feature vector is built by stacking all filtered pixels into one vector.

- The manifold is created by using $n = 50$ nearest neighbors of each of 1000 randomly sampled faces from the *training set*. The reason for this limitation is that Matlab® runs out of memory when a larger number of high-dimensional feature vectors is used.
- The dimensions corresponding to the $m = 50$ smallest eigenvalues are finally used for training the angle estimation with an `SVR`.
- The mapping from the high-dimensional feature space onto the manifold is also learned by a `GRNN` in order to allow head pose estimation for a new face image.

A deeper analysis of Figure 5.17 clearly shows that the performance on the *test set* is quite good as it originates from the same database as the *training set*. On a different database like the Siemens dataset the performance degrades enormously. One reason for this could be that database specific properties affect the manifold embedding so much that a correct mapping is not possible. Manifold Embedding therefore does not generalize well.

5.3.2.3 Discussion of the Results



(a) BME for the FacePix database

(b) ROC plot for frontal pose classification

Figure 5.18: Yaw-Angle encoding in the manifold embedding. The three dimensions corresponding to the strongest eigenvalues are displayed and all points are colored depending on the yaw angle they represent.

When the results of the implementation of Balasubramanian *et al.* (Figure 5.15) are compared to the adapted version in Figures 5.16 and 5.17, there is a huge difference in

pose estimation errors as well as frontal-pose detection. This has several reasons:

- When looking at the manifold embedding in Figure 5.18a it can be seen that the pose space is very discriminative due to the one-dimensional mapping of the yaw angle. This makes the GRNN and SVR training very easy and robust.

  In Figure 5.18b the two-dimensional mapping of pitch and yaw angles is shown. The manifold is no longer a single line but a grid of discrete angles, which are used in the training data. This makes both the GRNN mapping as well as SVR training a much harder task and introduces errors. What cannot be seen in this plot is the location of non-trained samples which are not embedded that smoothly into the manifold and often lie in between poses.

- FacePix images contain a uniform background. So no background clutter affects the creation of the manifold. In addition to that, there is a pose-related shadow which becomes visible on the background on certain yaw angles.

- As stated in Section 3.2.2, eye positions move systematically in the images depending on the yaw angle. This is a very strong hint in the feature vector which explains the good performance especially in non-frontal poses. In the BU-3DFE database there is no movement of the eyes as they are kept at a constant position across poses.

5.3.3 Conclusion

An HPES that uses BME shows a good pose estimation performance on a database which has homogeneous lighting and background. In the special case of the FacePix database, there are very little pose estimation errors due to the highly discriminative head images of different poses.

But as soon as background clutter is involved, the embedding is not able to maintain this accuracy. Part of this is due to how manifolds are created. As only Euclidean distances between feature vectors discriminate between images, cluttered backgrounds can affect that distance. Also, when pose angles with more than one DOF are used, the mapping becomes less discriminative.

The largest disadvantage that has been shown in the experiments is a weak generalization of the approach on databases that are different from the training database.

5.4 Hybrid Head Pose Estimation

Both the HOG/SVR system and BME method have their advantages. A combination of both approaches is therefore evaluated for a system where a biased Laplacian Eigenmap manifold embedding is performed on the HOG descriptor prior to SVR learning. The mapping between the HOG descriptor and the manifold is learned by a GRNN in order to map new descriptors onto an existing manifold.

5.4.1 Choice of Parameters

The same HOG descriptor described in Section 5.2.1 is used. As parameters for the LE mapping $n = 50$ nearest neighbors and $m = 50$ dimensions are selected. The spread of the GRNN is chosen relatively small at $\sigma = 0.4$ for this task. Finally, the SVR settings need to be adapted to the manifold and were determined to be best at $C = 4$ and $\gamma = 0.5$ at a tuning parameter $\nu = 0.01$.

5.4.2 Performance Evaluation

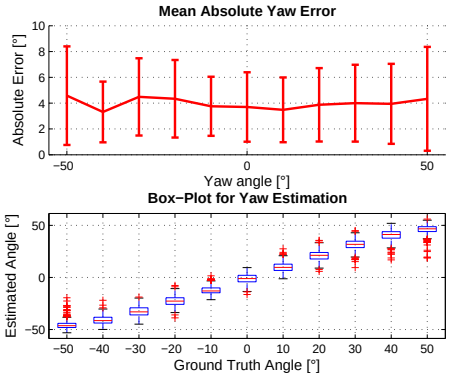
In Figures 5.19 and 5.20 the hybrid approach of HOG descriptors and manifold embedding is evaluated. Compared to the plain HOG descriptor without manifold embedding there is a decrease in performance. Also the boxplots show more outliers (red crosses). This is due to the fact that manifold embedding is generally not good for handling data that is not in the training set.

A positive effect of manifold embedding is the relative constant error rate across pose. When looking at the manifold structure in Figure 5.18b a grid structure becomes visible. This is because manifold embedding exploits the grid structure of the *training data* and moves data/pose points that do not belong to the same pitch/yaw angles farther away from each other. As a consequence, the effects of background clutter are reduced.

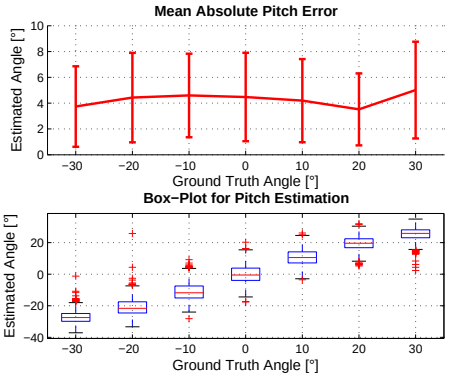
5.4.3 Performance on Additional Datasets

On additional databases (see Figure 5.21), a similar performance decrease for frontal pose classification similar to the Siemens database can be observed compared to the evaluations of the simple HOG/SVR method in Section 5.2.4.

In contrast to that, the head pose estimation performance on the Facepix database shows an improvement due to the almost constant mean absolute error across yaw

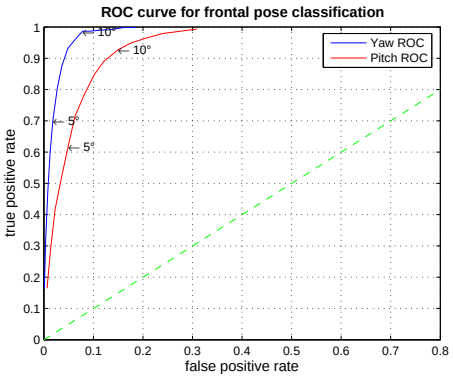


(a) Estimation performance for yaw angles ($E_y = 3.9, \sigma_y = 3.0$)

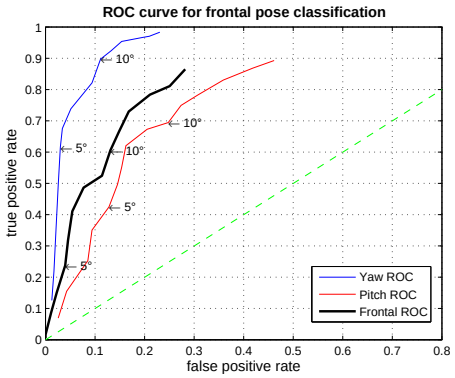


(b) Estimation performance for pitch angles ($E_p = 4.3, \sigma_p = 3.3$)

Figure 5.19: Performance of the HOG descriptor with manifold embedding on the test set



(a) ROC on the Testset



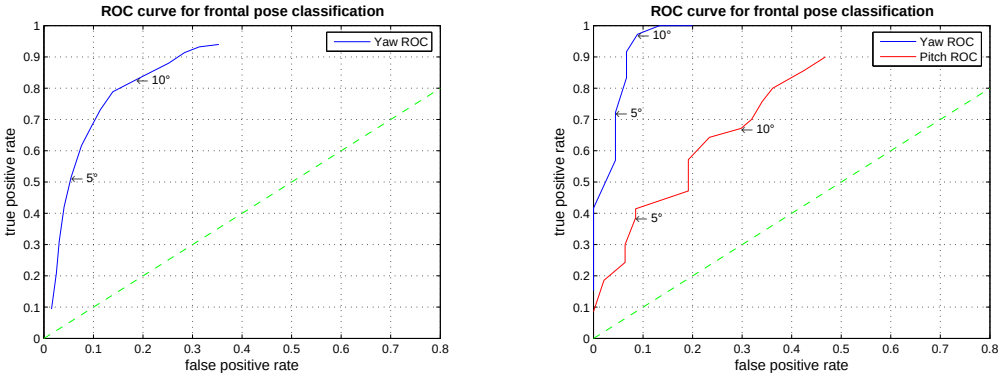
(b) ROC on Siemens dataset

Figure 5.20: ROC for the HOG descriptor with manifold embedding

angles (see Figure 5.22). Nevertheless, the frontal pose classification performance is decreased in a similar way as in the other databases due to a higher MAE at near frontal poses.

5.4.4 Conclusion

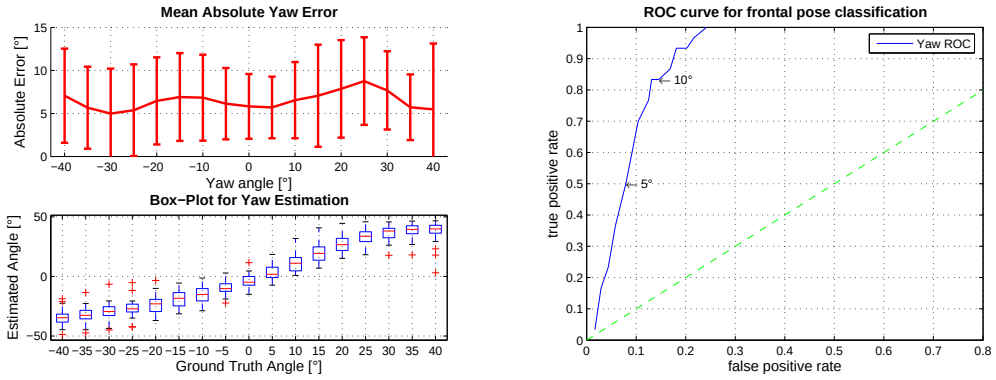
Using the HOG descriptor improves the performance of the BME approach by Balasubramanian *et al.* [1] significantly. The results are similar but slightly worse than the method



(a) ROC Performance on FERET

(b) ROC Performance on CMU PIE

Figure 5.21: ROC for the HOG descriptor with manifold embedding on additional databases



(a) Mean absolute errors for different yaw angles

(b) Receiver Operating Characteristics

Figure 5.22: Performance of the HOG descriptor with manifold embedding on the Facepix database

where an HOG descriptor output is mapped to angles directly by an SVR. The biggest advantage of the manifold embedding step is that an almost constant mean absolute pose estimation error can be achieved across pose. Unfortunately, the error at frontal poses is higher compared to the HPES without manifold embedding. This effect reduces the frontal pose classification performance, which has been shown on multiple databases.

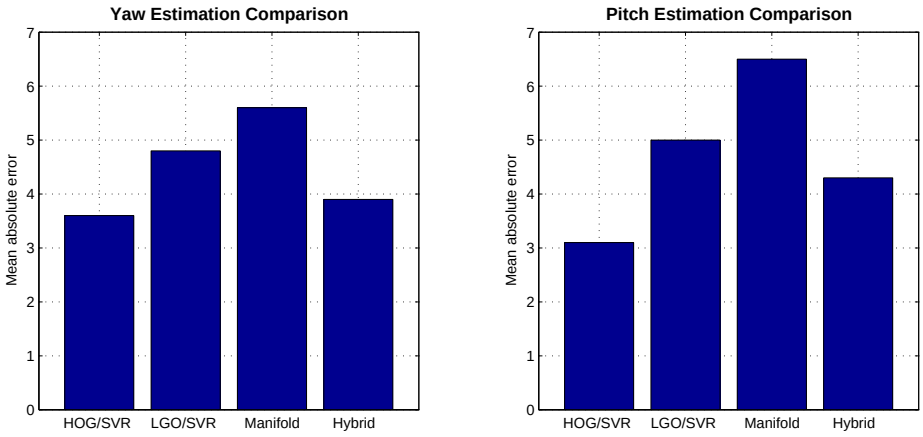


Figure 5.23: Comparison of the HOG, LGO, Manifold Embedding and Hybrid method

5.5 Comparison of the Approaches

This section compares the results of this thesis with each other. In addition to that, the performance of additional head pose estimation systems is compared with results obtained in this thesis.

5.5.1 HOG and BME Based Systems

The evaluations in the last sections have shown that a head pose estimation system benefits from the strengths of the `HOG` descriptor in combination with an `SVR` learning method. Using an `LGO` descriptor yields a lower head pose estimation performance. The `BME` method shows a good performance only in an one-dimensional yaw-pose estimation system on a special database but fails to generalize on the *test set* with pitch and yaw angle estimation. A hybrid approach that introduces a biased manifold embedding method in order to reduce the dimensionality of an `HOG` descriptor keeps the pose estimation error constant across pose. This improvement comes at the cost of a higher error near frontal poses which reduces the frontal pose classification performance.

The mean absolute pose estimation errors are summarized in Figure 5.23 for both pitch and yaw angle estimation on the *test set*. For both `DOF` the `HOG/SVR` method is superior to other methods presented in this thesis. Therefore the `HOG/SVR-HPES` proposed in Section 4.3 is recommended to be used in a real-world scenario. This has been shown by evaluations on several publicly available databases. The mean absolute error always stays well below ten degrees for both yaw and pitch estimations.

Table 5.2: Comparison of different algorithms in terms of mean absolute angle error of pitch and yaw estimations

Publication	Yaw Error	Pitch Error	Method
Pitch and Yaw estimation			
This work	**3.60** °	**3.10** °	HOG descriptor with SVR
Vatahska [55]	4.17 °	5.12 °	Features and boosting
Murphy-Chutorian [36]	4.67 °	3.39 °	LGO with SVR
Murphy-Chutorian [36]	4.82 °	5.00 °	own implementation
Yan [63]	6.72 °	8.87 °	Manifold Embedding
Murphy-Chutorian [34]	8.25 °	4.78 °	LGO descriptor with SVR
Voit [58]	8.50 °	12.50 °	Neural Network Regression
Stiefelhagen [48]	9.50 °	9.70 °	Neural Network Regression
Y. Li [28]	10.30 °	9.70 °	Eigenface with SVR
only Yaw estimation			
Fu [14]	1.70 °	–	Graph Embedding LLE
Balasubramanian [1]	3.57 °	–	BME with LE
Balasubramanian [1]	3.80 °	–	own BME implementation

ROC plots have shown that it is possible to detect the frontal pose with a true positive rate of 90% while maintaining a 10% false positive rate for yaw angles most of the time. Unfortunately these values cannot be achieved for the pitch angle estimation.

The reason for the lower pitch angle classification performance is most probably that image gradients change less distinctively in up-down pose changes. Also, the head images used to train the HPES do not show a real pose change but rather a rotation of the whole upper body.

5.5.2 Comparison to other Algorithms

There is a large number of other algorithms that deal with head pose estimation. In Section 2 various algorithms and methods that try to solve this task have been presented. Comparing the results of different approaches is a difficult task as almost all algorithms use different databases for training and have different outputs. As stated in [35], the most common comparison method is the mean absolute angular error for pitch and yaw estimations.

In this thesis only algorithms which continuously estimate both pitch and yaw angles in monocular single images are compared. In addition to that, some results for algorithms which estimate just yaw angles are given. This comparison can be found in Table 5.2.

It can be seen that the algorithm described in this thesis achieves the lowest mean error rates in its class. Other algorithms only achieve better results using multiple cameras, tracking methods or limited DOF. But as mentioned before, due to the lack of a unified benchmark and different databases the ranking of Table 5.2 must be interpreted with a certain caution.

5.6 Evaluation of Robustness

A real-world head pose estimation system (HPES) needs to be robust against various sources of influences. For example, the head images can have arbitrary backgrounds or lighting variations. In the following sections the pose estimation performance of the HOG descriptor in combination with an SVR is evaluated under such influences.

5.6.1 Influence of Random Backgrounds

Due to possible background clutter in a real-world scenario, the proposed HPES needs to handle the influence of random backgrounds. The most promising approach is to include backgrounds into the images of the *training set* and let the SVR algorithm learn to ignore their influence. Other methods for treating backgrounds, like background segmentation, often require video streams, uniform backgrounds or frontal poses. Therefore they are not applicable for the HPES presented in this thesis.

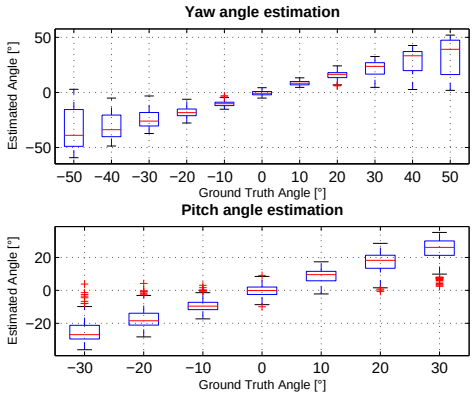
In order to evaluate the effectiveness of this approach, the system is trained with or without random backgrounds in face images. Figure 5.24 shows the pose estimation performances of the system on the *test set*, which contains both random and uniform backgrounds, under three different training conditions:

(a) trained on face images without backgrounds (only uniform color backgrounds)

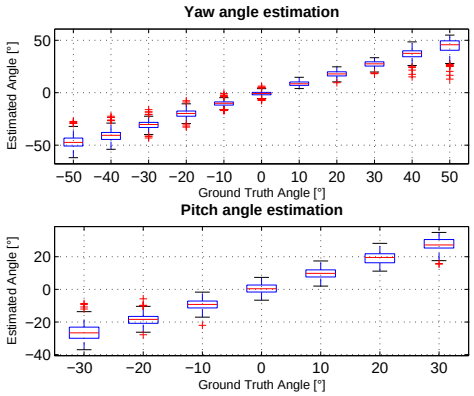
(b) trained on face images with random backgrounds only

(c) trained on face images with a mixture between random backgrounds and a uniform background

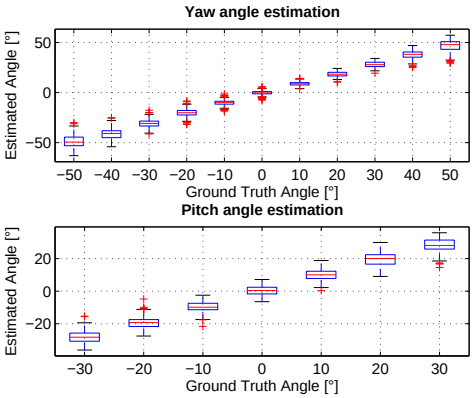
The mean absolute estimation error at larger yaw angles increases significantly when only a uniform background exists in training images. At these yaw angles, random backgrounds disturb the gradients of the orientation histograms when the system is tested on the *test set*. As soon as random backgrounds are included in the training step,



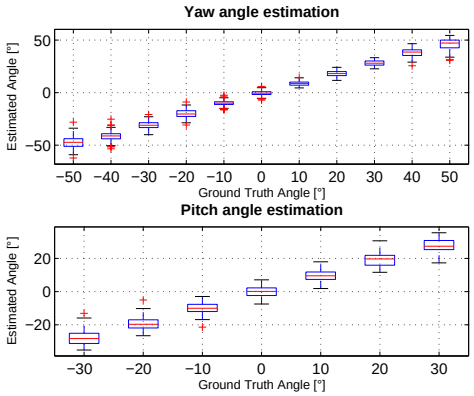
(a) Uniform backgrounds in training set ($E_y = 8.0, \sigma_y = 10.2$; $E_p = 4.4, \sigma_p = 4.7$)



(b) All training images with background ($E_y = 3.7, \sigma_y = 3.9$; $E_p = 3.2, \sigma_p = 2.7$)



(c) Mixed background for training and testing ($E_y = 3.1, \sigma_y = 3.1$; $E_p = 2.5, \sigma_p = 2.4$)



(d) Same training as in (c) but tested on random background only ($E_y = 3.1, \sigma_y = 2.9$; $E_p = 3.0, \sigma_p = 2.5$)

Figure 5.24: Influence of backgrounds in training data on the *test set* results.

this error is reduced. Therefore it is best to include a mixture of random backgrounds as well as uniform colors into the images of the *training set*. Even when the *test set* contains only images with random background, the error does not change significantly (see Figure 5.24d). Figure 5.25 shows three example images that all have a different background.

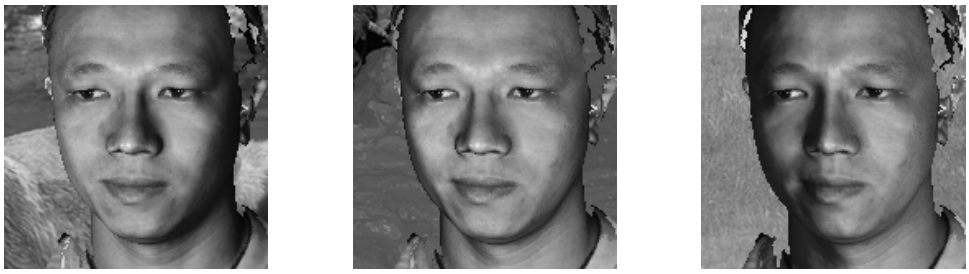


Figure 5.25: One face from the 3D-BUFE database at one pose with different lighting and backgrounds

5.6.2 Influence of Lighting

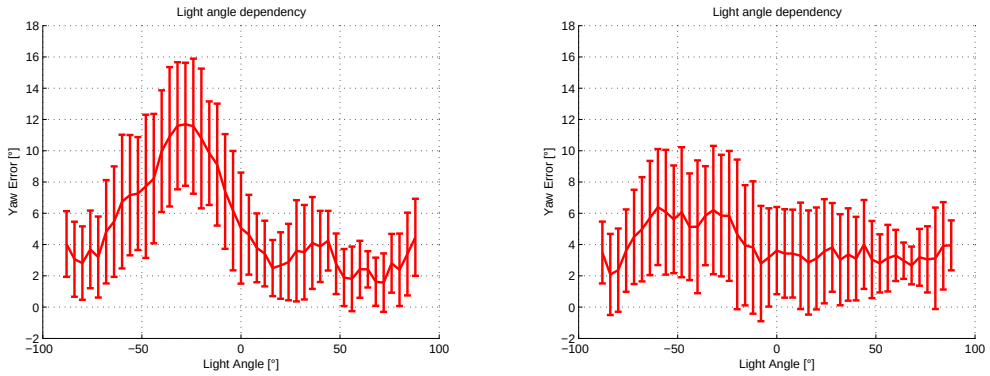
Non-uniform lighting can affect the accuracy of an HPES. While most global lighting effects are eliminated due to gradient-based pre-processing, shading does have a direct influence on image gradients.

Due to the lack of reliable lighting normalization techniques that both work at unknown poses and preserve all relevant gradients, the variation of lighting needs to be included as a part of the learning problem. In Figure 5.25 a series of three pictures with different lighting is shown which is generated for every face in the *training set* (see Section 3.3.4 for more details). Three lighting angles have proven to be a good compromise between robustness and the number of training images.

The effectiveness of this enhancement can be verified on the FacePix 30 database [4, 30] which contains a light variation set for each of the 30 subjects. A spot light is rotated around the head with a frontal pose in 1 ° increments. In Figure 5.26 the pose estimation performance is evaluated on these changing light angles. The error values visible in the plot are the mean absolute deviations from the 0 ° yaw angle of the faces.

Two experiments were performed: In the first experiment (Figure 5.26a) no lighting variation was used when training the system. The second experiment (Figure 5.26b) shows the light dependent pose errors when the system is trained using three different light angles for each face pose.

The performance of the detection stays around a MAE of 3.9 ° and is not dependent on the light angle when the training data contains lighting variation. This is not the case when only homogeneously lit training samples are used. In Figure 5.26a the estimation of the frontal pose shows a systematic error with a peak at approximately -30 ° lighting angle. This peak is caused by an asymmetry error in the recording of the moving-light



(a) without light variation in training data

(b) with light variation in training data

Figure 5.26: Frontal pose estimation error at changing light angles

image series where the rotating spot light is always brighter at negative light angles compared to positive angles.

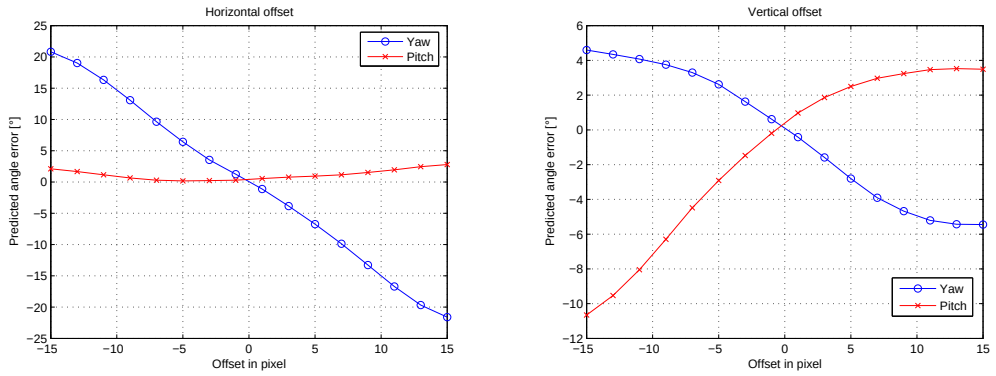
Therefore the proposed HPES with light variation training is sufficiently robust against variations in lighting of the face. No preceding light normalization is needed. The only step that is useful is gamma correction which ensures consistent gradient magnitudes in shadow and light areas [52].

5.6.3 Influence of Face Localization Accuracy

For a good reliability of the head pose estimation algorithm, all faces in the *training set* need to be aligned with respect to the position of the eyes. As stated in Section 4.3.1, there is no localization step included in the HPES. Such a step would introduce localization errors which would not allow an objective evaluation of the pose estimation performance alone. Both the *training* and *test set* as well as the Siemens database are aligned so that the eye position is constant in all images. In a complete head pose estimation system, a face detection step should perform this alignment.

The influence of an incorrect alignment of the head images is shown in Figure 5.27. The plots show the mean angle error for frontal poses of the *test set*. Either the horizontal or vertical component of the face position is modified by an offset with up to 15 pixels (at a descriptor size of 120 pixels or a cell size of 24 pixels respectively).

A good localization of the face is essential for correct pose estimation but an offset does not lead to an abrupt change in the estimated angles. It is not surprising that an



(a) Descriptor window is shifted horizontally

(b) Descriptor window is shifted vertically

Figure 5.27: Pose estimation error with inaccurate face localization

offset on the horizontal axis leads to a relative large change of yaw error considering that such a movement of facial structures is similar to a change in the left-right pose.

5.7 Runtime Measurements

A prototype of the HPES has been implemented in Matlab® together with some MEX files. The platform was Matlab® 7 R14 on an Intel Centrino Duo at 2×1.66 GHz and 2 GB RAM. All runtime performance results presented in this section are measured by using an HOG descriptor with SVR learning as described in Section 4.3.

The time required to estimate head poses depends on multiple factors:

- Size of the input image and how much it has to be enlarged or scaled down
- Color images must be converted to gray levels
- Size of the HOG descriptor and thus the dimension of the feature vector
- The type of kernel and parameters used for SVR learning as well as the number of support vectors used

In Table 5.3 typical runtimes for the algorithm are shown. Runtimes are based on the profiling tools and do not include the time for file-loading and color conversions of the images. There are two different types of measurements: single image processing only estimates the angles for one image while batch processing estimates the angles for

Table 5.3: Runtime performance of the head pose estimation system

Component	Processing/Image	
	single	batch
Calculation of HOG descriptor	47 ms	8.8 ms
Estimation of the angles	390 ms	36.5 ms
Total estimation time	437 ms	45.3 ms

several thousand images in a unified way. The large differences between single and batch processing is due to vectorization and code caching in Matlab® .

Training times cannot be expressed in a per-image value. Training is always done in batches. The standard training size used in this thesis consists of 12 000 descriptors which are precomputed before the training starts. The total training time is 250 seconds with the SVR parameters from Section 5.2.1.4.

5.8 Qualitative Evaluation of the Face Detection

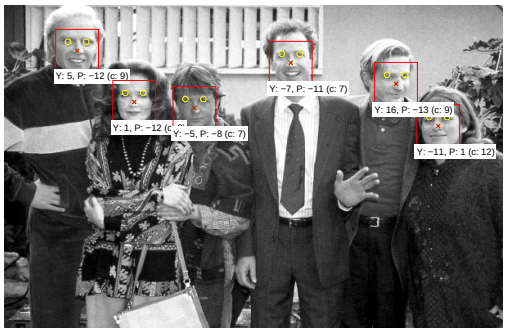
In Section 4.5 a simple face detection algorithm was proposed. This section shows a qualitative evaluation of the prototype. The SVM classifier is trained on 4000 randomly sampled images from the *training set* as well as 4000 arbitrary background patches.

For the classification task a linear SVM from the libsvm package [6] is used as it is able to not only output a binary decision but a confidence value for each classification. This facilitates the non-maximum suppression step and allows to reduce false detections.

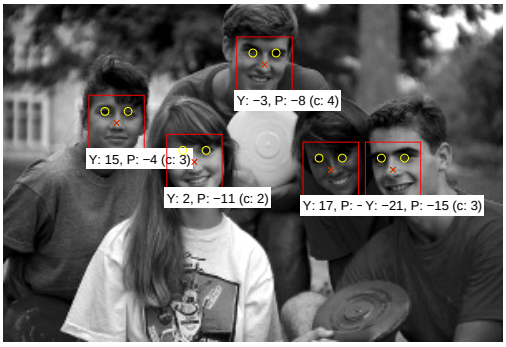
5.8.1 Face Detection Evaluations on Public Databases

In Figure 5.28 four images from the CMU Frontal dataset [42] are used to demonstrate the detection performance. Most faces were found in a sliding window fashion with a fixed scale which was defined by the eye distance of one of the subjects in the image. For each detected face, the pose estimation (*Y*aw and *P*itch) as well as the classification confidence (*C*) is shown. In Figure 5.28d problems with faces with eye-glasses (which result in misses) and in dark regions (which produce false positives) are shown. Including eye-glasses in the *training set* would reduce the first problem.

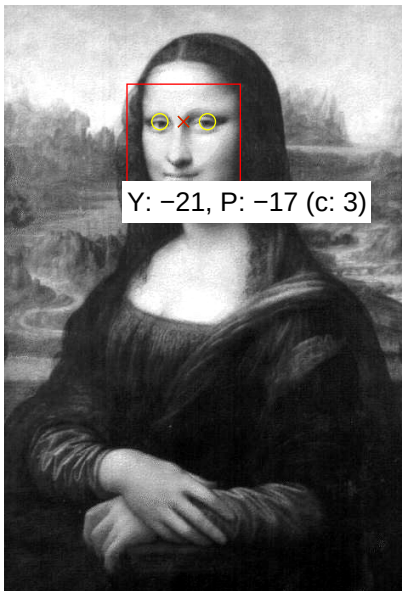
Additional results can be found in Figure 5.29 where the face detection algorithm successfully detected all 10 faces from a collection of images from the Yale Face Database B [15]. In addition to that the poses are correctly estimated within a 5 degree



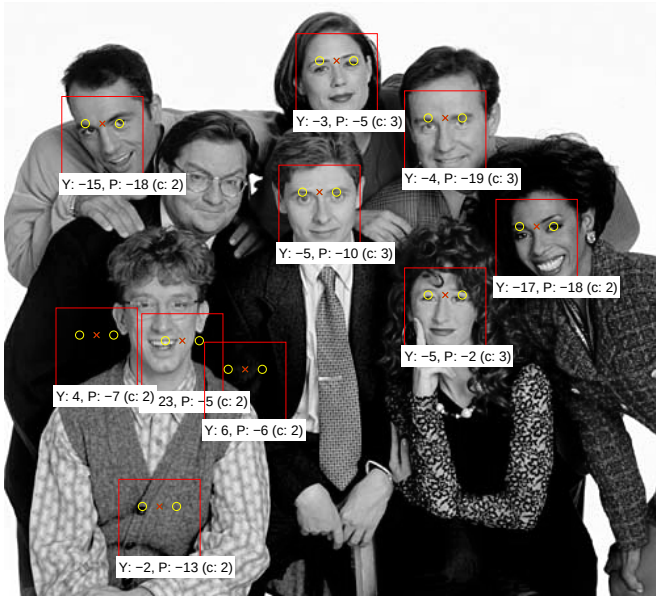
(a) file bttf301



(b) file frisbee



(c) file mona-lisa



(d) file newsradio

Figure 5.28: Face detection on example images from the CMU Frontal dataset

deviation from the frontal pose in all except one face (bottom, left) which has a slight upwards pose.

The results are not perfect but show a very low false positive rate considering that several thousand windows are evaluated per image. Note that no post-processing except a non-maxima suppression of the detected results is performed. One weak spot is the exact localization of the eyes because sometimes an offset of several pixels can occur. Often, eyebrows will deviate the face region in the vertical position. Therefore the face

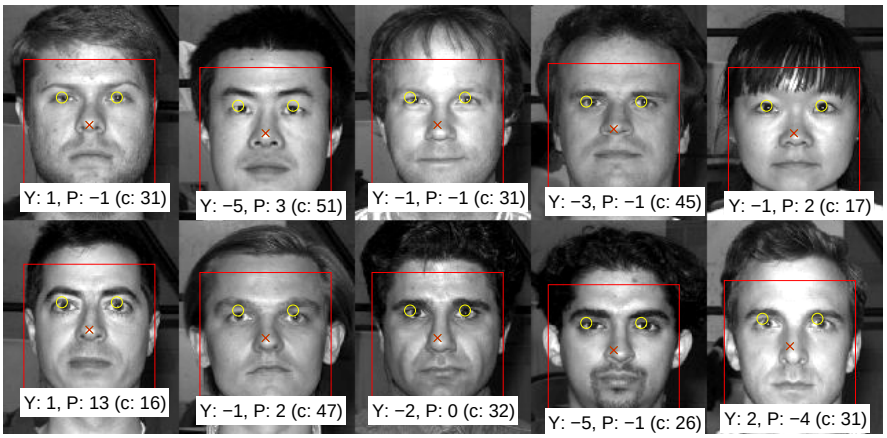


Figure 5.29: Face detections in a collection of faces from the Yale Face Database B

detection algorithm will introduce errors in pose estimation if the direct detection result is used as the only face detector.

Also, the runtime performance is rather slow. In the current Matlab® implementation the detection of all faces in a 640×311 pixel image like in Figure 5.29 takes around 50 seconds for extracting the descriptor in 13359 sliding windows and 16 additional seconds to classify each window into a face or non-face (or 5 ms per window position). A multi-view face detector using Haar features and a decision tree [56] can process an image of this size in less than one second.

5.8.2 FacePix Evaluations using Face Detection

This subsection will show how the face detection algorithm from Section 4.5 can be used to improve the estimation results on the FacePix database [4, 30]. As stated in Section 3.2.2, the registration of the eye positions is not very accurate across poses. This is why face detection can improve pose estimation compared to uncorrected FacePix images.

Face detection is done only along a horizontal line at the level of the eyes because FacePix images are normalized so that the eyes always have the same vertical position. Figure 5.30 shows one face detection where the detection rectangle is perfectly aligned with the eyes. The plot next to the image shows the face detection score of the shifted rectangle as it is shifted along the horizontal line at the level of the eyes (the x-

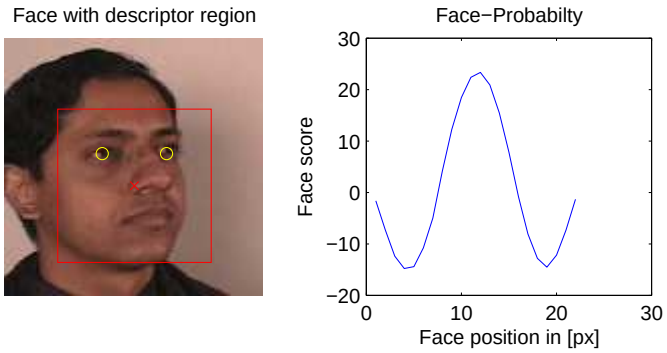
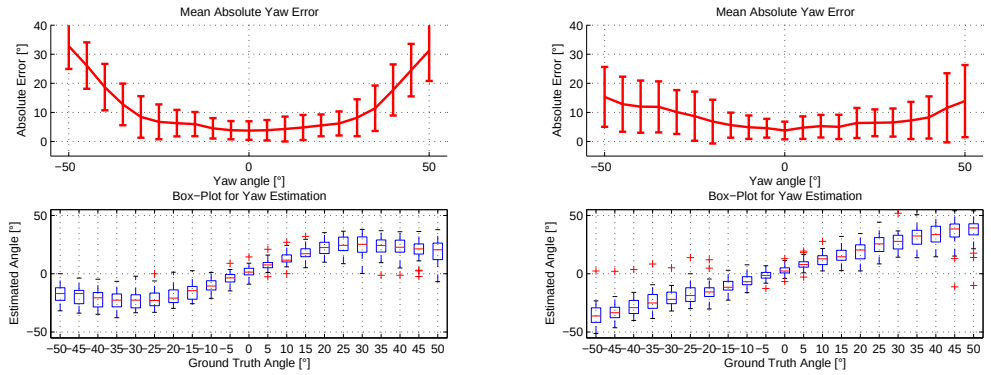


Figure 5.30: Face detection for one image of the FacePix database

coordinates of the descriptor are measures from its left-upper corner). The curve yields one maximum at the most likely face position.



(a) Performance with static face positions at the center of the image

(b) Performance with dynamic face detection which defines the position of each image individually

Figure 5.31: Pose estimation error comparison on the FacePix database

In Figure 5.31 a comparison of the head pose estimation accuracy between the original FacePix face images and dynamic face detection is shown. In the original FacePix images the face is assumed to have the same, static position in all images.

It can be seen that the mean absolute error in the static case (Figure 5.31a) shows an increase in the error at non-frontal poses. For the face detection case (Figure 5.31b), the error plot is a lot smoother as the pose related eye shift of non-frontal poses is tracked by the detection window. Note that a small error is introduced by inaccurate localization of the face which results in a larger variance of the absolute error.

5.8.3 Conclusion

This section presented a method to detect multi-pose faces in an image. Using an SVM algorithm that can classify HOG descriptors of a sliding window is sufficient to detect human heads with a high certainty.

Unfortunately, this method is very slow and the detections do not always detect the face without an offset. Also, there already exist well developed systems for face detection in multi-pose environments [56] that perform significantly faster. But as stated in Section 4.5, this implementation has many possibilities for improvements. It could be worth to be investigated in future work.

5.9 Summary and Discussion

The chapter about experimental results presented a thorough evaluation of the head pose estimation system (HPES) proposed in Chapter 4. At the beginning, the description of the evaluation setup was given. This setup was used to evaluate the pose estimation performance of three different algorithms with an HOG descriptor, a manifold embedding step and a combination of both techniques.

The evaluation has shown that the pose estimation accuracy is best when the HOG descriptor is used and the mapping between the descriptor vectors and pose angles is learned by a ν-SVR machine. Any intermediate steps, like manifold embedding of descriptor values, did not improve the performance. Contrary to what one might expect from the results in [1], manifold embedding of the LoG transformed images alone lead to a bad performance when more than one DOF was estimated. Therefore the use of the HOG descriptor with SVR learning is recommended for head pose estimation.

When the HOG/SVR method presented in this thesis is compared to the results in other publications, it turned out to achieve the best performance in its class. While this result has been achieved on a specialized database for head pose estimation, similar results were obtained on publicly available databases.

An analysis of the pose estimation error has shown that the accuracy increases in near-frontal poses. This makes the HOG/SVR-algorithm ideal for frontal pose detection. Yaw classification consistently outperformed pitch classification though.

In addition to the performance evaluation the influence of lighting and incorrect face localization was investigated. The pose estimation performance of the system on a database that has non-uniform light angles significantly increased when light variations

were included in the training set. Even though there is an infinity of possibilities for lighting variations, the use of three light angles for each pose seams to be sufficient in the training database.

A good face localization is essential when a high accuracy of the predicted pose angles is required. Fortunately, small deviations from the optimal descriptor position do not cause a large error when an HOG descriptor is used.

The chapter concluded with a qualitative demonstration of the HOG based face detector and a possible application. Unfortunately a slow runtime in connection with offsets in the detections leads to the conclusion that this method is inferior to existing algorithms. Nevertheless, there is a lot of space for improvements in both speed as well as accuracy.

6 Conclusion

In this thesis a system for head pose estimation from monocular still images was presented. After a thorough evaluation of existing HPES approaches, a method which promises low errors in real world scenarios was chosen as a basis for this work. The main idea of this approach is to extract gradient orientations of a face patch and let a non-linear regressor learn the mapping from these orientations to pose angles. The structure of the original algorithm was kept, while a better descriptor, the Histogram of Oriented Gradients, was introduced to describe the facial patch. The high dimensionality of this descriptor required the use of two support vector machines with RBF kernels in order to solve the regression task efficiently. The SVR machines are trained to estimate either the pitch or the yaw angle of a human face while they are robust towards lighting and background clutter.

A head pose estimation system consists not only of training a machine learning algorithm. It requires a thoughtfully designed database which provides the training data. As no publicly available database was able to provide either enough variance in poses, lighting or backgrounds, a new database was created which renders 3D laser scans of human heads in multiple poses and lighting conditions. In addition to that, arbitrary backgrounds were combined with the head images in order to make the system more robust. A pose increment of 10 degrees in either yaw or pitch angle is sufficient to train an HPES. It is essential to include a large variety of images of different individuals to ensure the person invariance of the system.

A subset of this custom database was used to train the HPES. The performance of the system was then evaluated on another subset of this database, the *test set*. The main goal of this thesis was to estimate the pose angle of an arbitrary given face image with

an absolute error of less than 10 degrees and to classify a frontal pose accurately. On the *test set* that goal has been achieved without problems and an accuracy of less than four degrees in both pitch and yaw angle estimation was demonstrated. Also, the frontal pose classification is possible with a 90% success rate at less than 5% false detections when a threshold of 5°, as required by the ICAO standard, is used. The optimal classification performance is achieved with a threshold of around 7° though.

On additional databases a similar performance was shown for yaw angles. Unfortunately, for pitch angles the HPES could not maintain this performance. A better training database using real faces could therefore improve this performance.

In addition to the HOG/SVR based head pose estimation approach, a second HPES which uses a manifold embedding algorithm was implemented. It could however not achieve the good results of the gradient orientation based system.

As HOG descriptors have proven to be useful in tasks like pedestrian detection, a face detection algorithm was presented which uses the same HOG descriptor as for the pose estimation task in order to find faces in arbitrary images. In a qualitative evaluation it has been shown that HOG descriptors in combination with a linear SVM classification are suited for this task. A very low false positive rate could be achieved while almost all faces in the test images were found. Some post-processing might even improve the localization accuracy which sometimes shows an offset error of a few pixels from the optimal position.

In short, it can be said that an HOG based head pose estimation system with a suitable training database is able to reliably estimate poses in two DOF with a decent performance. This has been proven on a custom head pose database as well as on publicly available databases.

6.1 Future Work

The head pose estimation system presented in this thesis is able to achieve state of the art results in terms of error rates and robustness. Yet there is still enough room for improvements.

Better training data: Using synthetic data has proven to yield good results even on real world datasets. Yet an improvement in performance can be expected when a dataset consisting of real world head pose data is used. An angular grid resolution of about 10° in both pitch and yaw angles should be sufficient. Also, a rather

small number of about 20 to 30 subjects should be enough to train a good system. When collecting training data, a wide range of facial features like glasses, facial hair, expressions and so on should be present to make the final system even more robust and reliable.

Descriptor: The current HOG descriptor weights each gradient in the facial patch equally strong. Due to hair styles and background clutter gradients near the border of the face contain more noise than gradients closer to the face center. Therefore the effect of a center weighting method for gradients in the histogram calculation should be investigated.

Regression: Support vector regression does a very good job in estimating yaw and pitch angles. One downside is that it estimates these two angles individually. A regression algorithm that features the sparse vector and kernel advantages of the SVR but is able to handle multiple output dimensions simultaneously could exploit synergy effects between both angles in combined poses. Therefore an even better estimation performance may be possible.

Pose classification: The algorithms presented in this thesis implement pose classification as a threshold of the continuous angle estimation. There exist algorithms that solve this task directly via a classification of a face image into discrete poses. Maybe using such an algorithm achieves better results for the detection of the frontal pose.

Face localization: Most testing data is normalized by a face tokenization algorithm [49]. This algorithm only performs reliable in a very narrow range around the frontal pose. A face localization that works better in non-frontal poses while maintaining the same accuracy can therefore enhance the pose estimation performance. The face detection algorithm presented in this thesis solves this task to a certain extent. Still, many improvements in both speed and accuracy would be required in order to make it useful in a head pose estimation system.

This enumeration of possible improvement shows that the head pose estimation presented in this thesis has a large potential for improvements while maintaining its main elements. It is therefore reasonable to invest future work into this system and maximize its strengths.

A

Acronyms and Symbols

List of Acronyms

AAM	Active Appearance Models
BME	Biased Manifold Embedding
BU-3DFE	Binghamton University 3D Facial Expression
CMU	Carnegie Mellon University
CUbiC	Center for Cognitive Ubiquitous Computing
DOF	degrees of freedom
DoG	Difference of Gaussian
EM	Expectation Maximization
FERET	Face Recognition Technology
GRNN	Gaussian Regression Neural Network
HOG	Histogram of Oriented Gradients
HPES	Head Pose Estimation System
LE	Laplacian Eigenmaps
LDA	Linear Discriminant Analysis
LGO	Localized Gradient Orientation
LLE	Locally Linear Embedding
LoG	Laplacian of Gaussian
MAE	Mean Absolute Error
ME	Manifold Embedding
MEX	Matlab® Executable

MRTD	Machine Readable Travel Documents
PCA	Principal Component Analysis
PIE	Pose Illumination Expression
RBF	Radial Basis Function
ROC	Receiver Operating Characteristics
ROI	region of interest
RVM	Relevance Vector Machine
SIFT	Scale Invariant Feature Transform
SVR	Support Vector Regression
SVM	Support Vector Machine
TFIT	Token Face Image Type
VRML	Virtual Reality Markup Language

List of Symbols

$\langle x, y \rangle$	Tensor dot product
$\lfloor . \rfloor$	Floor operator
$\lvert . \rvert$	Absolute value
$x * y$	Convolution
∇	Nabla operator
$\triangle$	Laplace operator
$\wedge$	Logical AND
$\vee$	Logical OR
$\neg$	Logical NOT

Bibliography

[1] Balasubramanian, V. N., Krishna, S., and Panchanathan, S. (2008). Person-independent head pose estimation using biased manifold embedding. *EURASIP Journal on Advances in Signal Processing*, 2008:1–15.

[2] Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computing*, 15(6):1373–1396.

[3] Beymer, D. (1994). Face recognition under varying pose. In *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition*, pages 756–761.

[4] Black, J., Gargesha, M., Kahol, K., Kuchi, P., and Panchanathan, S. (2002). A framework for performance evaluation of face recognition algorithms. In *Proceedings of Internet Multimedia Management Systems III*, volume 4862. ITCOM. `http://www.facepix.org/`.

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

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

[7] Choi, K. N., Carcassoni, M., and Hancock, E. R. (1998). Estimating 3D facial pose using the EM algorithm. In *Face Recognition: From Theory to Applications*, pages 412–423. Springer.

[8] Cootes, T., Edwards, G., and Taylor, C. (1998). Active appearance models. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 1407, pages 484–498. Springer.

[9] Cootes, T., Walker, K., and Taylor, C. (2000). View-based active appearance models. In *Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition*, pages 227–232.

[10] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886–893.

[11] Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A. J., and Vapnik, V. (1996). Support vector regression machines. In *Neural Information Processing Systems*, pages 155–161.

[12] Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874.

[13] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

[14] Fu, Y. and Huang, T. S. (2006). Graph embedded analysis for head pose estimation. In *Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition*, pages 3–8.

[15] Georghiades, A., Belhumeur, P., and Kriegman, D. (2001). From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660. `http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html`.

[16] Gonzalez, R. and Woods, R. (2002). *Digital Image Processing*. Prentice-Hall, 2nd edition.

[17] Gourier, N., Hall, D., and L., C. J. (2004). Estimating face orientation from robust detection of salient facial features. In *Proceedings of Pointing 2004, International Workshop on Visual Observation of Deictic Gestures*, pages 17–25, Cambridge, UK. `http://www-prima.inrialpes.fr/Pointing04/`.

[18] Gross, R., Matthews, I., and Baker, S. (2004). Constructing and fitting active appearance models with occlusion. In *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshop*, volume 5, page 72.

[19] Gui, Z. and Zhang, C. (2006). 3D head pose estimation using non-rigid structure-from-motion and point correspondence. In *Proceedings of the IEEE Region 10 Conference*, pages 1–4.

[20] Guo, G., Fu, Y., Dyer, C., and Huang, T. (2008). Head pose estimation: Classification or regression? In *Proceedings of the 19th International Conference on Pattern Recognition*, pages 1–4.

[21] Hsu, C.-W., Chang, C.-C., and Lin, C.-J. (2009). *A Practical Guide to Support Vector Classification*. Department of Computer Science, National Taiwan University. `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.

[22] Huang, J., Shao, X., and Wechsler, H. (1998). Face pose discrimination using support vector machines (SVM). In *Proceedings of the 14th International Conference on Pattern Recognition*, pages 154–156.

[23] ISO/IEC 14772-1:1997 (1997). Virtual reality modeling language. `http://www.web3d.org/x3d/specifications/vrml/`.

[24] ISO/IEC 19794-5:2005 (2005). Information technology – biometric data interchange formats – part 5: Face image data.

[25] Jiang, W. (2007). *Human Feature Extraction in VS image Using HOG Algorithm*. Center for Biometrics and Security Research, Chinese Academy of Sciences, Beijing. `http://home.ustc.edu.cn/~jiangwei/HumanFeatureExtractioninVSimageUsingHOGAlgorithm.pdf`.

[26] Keerthi, S. S. and Lin, C.-J. (2003). Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Computing*, 15(7):1667–1689.

[27] Krüger, V. and Sommer, G. (2002). Gabor wavelet networks for efficient head pose estimation. *Image and Vision Computing*, 20(9):665–672.

[28] Li, Y., Gong, S., and Liddell, H. (2000). Support vector regression and classification based multi-view face detection and recognition. In *Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition*, page 300.

[29] Li, Z., Fu, Y., Yuan, J., Huang, T. S., and Wu, Y. (2007). Query driven local linear discriminant models for head pose estimation. In *Proceedings of the IEEE International Conference on Multimedia & Expo, Beijing, China*.

[30] Little, D., Krishna, S., Black, J., and Panchanathan, S. (2005). A methodology for evaluating robustness of face recognition algorithms with respect to variations in pose angle and illumination angle. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 89–92.

[31] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110.

[32] Ma, Y., Konishi, Y., Kinoshita, K., Lao, S., and Kawade, M. (2006). Sparse bayesian regression for head pose estimation. In *Proceedings of the 18th International Conference on Pattern Recognition*, pages 507–510.

[33] McKenna, S. J. and Gong, S. (1998). Real-time face pose estimation. *Real-Time Imaging*, 4(5):333–347.

[34] Murphy-Chutorian, E., Doshi, A., and Trivedi, M. (2007). Head pose estimation for driver assistance systems: A robust algorithm and experimental evaluation. In *Proceedings of the IEEE Conference on Intelligent Transportation Systems*, pages 709–714.

[35] Murphy-Chutorian, E. and Trivedi, M. (2009). Head pose estimation in computer vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):607–626.

[36] Murphy-Chutorian, E. and Trivedi, M. M. (2008). Hyhope: Hybrid head orientation and position estimation for vision-based driver head tracking. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 512–517.

[37] Ohayon, S. and Rivlin, E. (2006). Robust 3D head tracking using camera pose estimation. In *Proceedings of the 18th International Conference on Pattern Recognition*, pages 1063–1066.

[38] Osadchy, M., Cun, Y. L., and Miller, M. L. (2007). Synergistic face detection and pose estimation with energy-based models. *Journal of Machine Learning Research*, 8:1197–1215.

[39] Phillips, P. J., Wechsler, H., Huang, J., and Rauss, P. J. (1998). The FERET database and evaluation procedure for face-recognition algorithms. *Image and Vision Computing*, 16:295–306. `http://www.itl.nist.gov/iad/humanid/feret/`.

[40] Raytchev, B., Yoda, I., and Sakaue, K. (2004). Head pose estimation by nonlinear manifold learning. In *Proceedings of the 17th International Conference on Pattern Recognition*, volume 4, pages 462–466.

[41] Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

[42] Rowley, H. A., Baluja, S., and Kanade, T. (1998). Rotation invariant neural network-based face detection. In *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition*, page 38.

[43] Seemann, E., Nickel, K., and Stiefelhagen, R. (2004). Head pose estimation using stereo vision for human-robot interaction. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, pages 626–631.

[44] Sherrah, J., Gong, S., and Ong, E.-J. (1999). Understanding pose discrimination in similarity space. In *Proceedings of the 10th British Machine Vison Conference*, pages 523–532.

[45] Sim, T., Baker, S., and Bsat, M. (2003). The CMU pose, illumination, and expression database. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1615–1618.

[46] Smith, W. A. P. and Hancock, E. R. (2008). Recovering face shape and reflectance properties from single images. In *Proceedings of the 8th IEEE International Conference on Automatic Face & Gesture Recognition*, pages 1–8.

[47] Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222.

[48] Stiefelhagen, R. (2004). Estimating head pose with neural networks – results on the Pointing04 ICPR workshop evaluation data. In *Proceedings of Pointing 2004, International Workshop on Visual Observation of Deictic Gestures*.

[49] Storer, M., Urschler, M., Bischof, H., and Birchbauer, J. A. (2008). Face image normalization and expression/pose validation for the analysis of machine readable travel

documents. In *Proceedings of the 32nd Workshop of the Austrian Association for Pattern Recognition on Challenges in the Biosciences: Image Analysis and Pattern Recognition Aspects*, volume 32, pages 29–39.

[50] Sung, J., Kanade, T., and Kim, D. (2008). Pose robust face tracking by combining active appearance models and cylinder head models. *International Journal of Computer Vision*, 80(2):260–274.

[51] Sung, J. and Kim, D. (2008). Pose-robust facial expression recognition using view-based 2D + 3D AAM. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 38(4):852–866.

[52] Tan, X. and Triggs, B. (2007). Enhanced local texture feature sets for face recognition under difficult lighting conditions. In *Analysis and Modeling of Faces and Gestures*, volume 4778, chapter 13, pages 168–182. Springer Berlin / Heidelberg.

[53] Tipping, M. E. (2001). Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244.

[54] Tukey, J. W. (1977). *Exploratory Data Analysis*. Addison-Wesley.

[55] Vatahska, T., Bennewitz, M., and Behnke, S. (2007). Feature-based head pose estimation from images. In *Proceedings of the IEEE-RAS 7th International Conference on Humanoid Robots*.

[56] Viola, P. and Jones, M. J. (2003). Fast multi-view face detection. Technical Report 096, Mitubishi Electronic Research Laboratories. `http://www.merl.com/reports/docs/TR2003-96.pdf`.

[57] Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154.

[58] Voit, M., Nickel, K., and Stiefelhagen, R. (2008). Head pose estimation in single- and multi-view environments – results on the CLEAR'07 benchmarks. In *Multimodal Technologies for Perception of Humans*, volume 4625 of *Lecture Notes in Computer Science*, pages 307–316. Springer.

[59] Wang, J.-G. and Sung, E. (2007). EM enhancement of 3D head pose estimated by point at infinity. *Image and Vision Computing*, 25(12):1864–1874.

[60] Wilson, H. R., Wilkinson, F., Lin, L.-M., and Castillo, M. (2000). Perception of head orientation. *Vision Research*, 40(5):459–472.

[61] Wollaston, W. H. (1824). On the apparent direction of eyes in a portrait. *Philosophical Transactions of the Royal Society of London*, 114:247–256.

[62] Wu, J. and Trivedi, M. M. (2008). A two-stage head pose estimation framework and evaluation. *Pattern Recognition*, 41(3):1138–1158.

[63] Yan, S., Zhang, Z., Fu, Y., Hu, Y., Tu, J., and Huang, T. (2008). Learning a person-independent representation for precise 3D pose estimation. In *Multimodal Technologies for Perception of Humans*, volume 4625 of *Lecture Notes in Computer Science*, pages 297–306. Springer.

[64] Yin, L., Wei, X., Sun, Y., Wang, J., and Rosato, M. J. (2006). A 3D facial expression database for facial behavior research. In *Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition*, pages 211–216.

[65] Zhao, Q., Zhang, D., and Lu, H. (2005). Supervised LLE in ICA space for facial expression recognition. In *Proceedings of the International Conference on Neural Networks and Brain*, volume 3, pages 1970–1975.

[66] Zhu, Q., Yeh, M.-C., Cheng, K.-T., and Avidan, S. (2006). Fast human detection using a cascade of histograms of oriented gradients. In *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition*, pages 1491–1498.