Lip Contour Localization using Statistical Shape Models
Fabian Schneiter

Master Thesis
Supervised by Gabriele Fanelli
Computer Vision Institute,
Department of Electrical Engineering & Information Technology,
ETH Zürich, Switzerland
Summer 2009
Abstract

This master thesis focuses on human lip contours localization in still images and video sequences depicting talking people in nearly-frontal view. Among other applications, knowing the lips accurate boundaries is desired in audio-visual speech recognition, which uses visual features to complement audio information, especially when the audio channel is affected by noise. We present a system comprising a cascade of different algorithms, from face detection to detailed lip contour localization and tracking. This thesis focuses on the latter, for which we employ Active Shape Models (ASM).

We analyze different approaches for fitting an ASM to an unseen image, aiming for increased lip contour localization accuracy and robustness. The most significant variations include the use of a novel mouth localization method to initialize the ASM search, the introduction of match weights, shape interpolation, and the use of 2D color eigenpatches for local appearance matching. Finally, some essential problems of lip tracking with ASMs are tackled and discussed.
Acknowledgements

I would like to thank my supervisor Gabriele Fanelli, who always greatly supported me during this project, gave me new ideas and often encouraged me to dig a little deeper.

Also, I would like to thank the following:

# List Of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM</td>
<td>Active shape model</td>
</tr>
<tr>
<td>AVSR</td>
<td>Audio-visual speech recognition</td>
</tr>
<tr>
<td>CUAVE</td>
<td>Clemson university audio-visual experiments, a database name</td>
</tr>
<tr>
<td>GLDM</td>
<td>Gray-level distribution model</td>
</tr>
<tr>
<td>LLASM</td>
<td>Lip localization active shape models, our implementation of the ASM algorithms (Matlab)</td>
</tr>
<tr>
<td>mROI</td>
<td>Mouth region of interest</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal component analysis</td>
</tr>
<tr>
<td>PDM</td>
<td>Point distribution model</td>
</tr>
<tr>
<td>RGB</td>
<td>Red-green-blue model, typical image color channels</td>
</tr>
<tr>
<td>ROI</td>
<td>Region of interest</td>
</tr>
<tr>
<td>STASM</td>
<td>Steve’s active shape models, a reference implementation of ASM algorithms (C++)</td>
</tr>
<tr>
<td>std</td>
<td>Standard deviation, a Matlab function name, used as a general data variance indicator</td>
</tr>
<tr>
<td>TASM</td>
<td>Train active shape models, the training program part of STASM</td>
</tr>
<tr>
<td>VJ</td>
<td>Viola-Jones, a common face detection algorithm</td>
</tr>
</tbody>
</table>
## Contents

1 Introduction .................................................. 11
   1.1 Motivation ............................................... 11
   1.2 Method .................................................. 12
   1.3 Architecture Overview ................................ 12
   1.4 Layout of this Report ................................. 13

2 Related Work ................................................. 14
   2.1 Lip Localization Techniques ............................ 14
   2.2 Lip Localization with Active Shape Models ............ 14

3 Active Shape Models ........................................ 16
   3.1 Overview ................................................ 16
   3.2 ASM Training .......................................... 17
      3.2.1 The Shape Model .................................. 17
      3.2.2 The Profile Model ................................. 19
   3.3 ASM Search ............................................ 22
      3.3.1 The Initialization Method ......................... 25
      3.3.2 Pyramid Level Search ............................. 25

4 Data Acquisition and Model Evaluation ..................... 27
   4.1 Training Data .......................................... 27
      4.1.1 Requirements ...................................... 27
      4.1.2 Existing Databases ................................ 27
      4.1.3 The CUAVE56-L Database .......................... 28
   4.2 Model Evaluation ...................................... 29
      4.2.1 Partitioning the Data Set ......................... 29
      4.2.2 Error Measurement - \( m_e \) ........................ 31
      4.2.3 Overall Quality Measure of Multiple Samples .... 31
   4.3 Reference Values - STASM & LLASM ................. 32

5 Model Variations & Parameters ............................ 34
   5.1 Overview and Variation Classification ................. 34
   5.2 Initialization Variations .............................. 35
   5.3 Shape Model Variations ............................... 36
5.3.1 Alternative Shape Sample Alignment ................................................. 36
5.3.2 Using Polar Coordinates ................................................................. 38
5.3.3 Shape Interpolation ................................................................. 38
5.4 Profile Model Variations ................................................................. 39
5.4.1 2D Profiles ............................................................................... 39
5.4.1.1 Milborrow Patches ................................................................. 40
5.4.1.2 PCA Patches ........................................................................ 40
5.4.1.3 Gabor Patches .................................................................... 43
5.4.2 Image Scale Normalization ............................................................... 45
5.4.3 Color vs. Grayscale Profiles ............................................................ 45
5.5 Algorithm Variations ........................................................................... 46
5.5.1 Match Weights ............................................................................. 46
5.5.2 Number of Eigenvectors on Different Pyramid Levels ...................... 47
5.5.3 Image Preprocessing ...................................................................... 48
5.6 General Parameter Variations .............................................................. 48
6 Tracking ............................................................................................. 50
6.1 Shape Filtering ................................................................................. 50
6.2 Using the Output-Shape of the Previous Frame .................................... 51
7 Experimental Results ........................................................................ 54
7.1 Results of Improvement Approaches Testing ...................................... 54
7.1.1 Initialization Variation Experiments .............................................. 54
7.1.2 Training Sample Alignment Variation Experiments ....................... 55
7.1.3 Polar Coordinates Experiment ..................................................... 56
7.1.4 Shape Interpolation Experiments ................................................. 56
7.1.5 2D Profile Comparison Experiments ........................................... 57
7.1.6 Image Scale Normalization Experiment ....................................... 58
7.1.7 Color-Grayscale Comparison Experiments .................................... 58
7.1.8 Match Weights Experiments ....................................................... 59
7.1.9 Number of Eigenvectors on Different Pyramid Levels Experiments .... 60
7.1.10 Image Preprocessing Experiment .............................................. 60
7.2 Combination of Improvement Approaches ........................................ 61
7.3 Tracking Results ............................................................................... 63
7.3.1 Filtering Experiment ..................................................................... 63
7.3.2 Using the Output-Shape of the Previous Frame .............................. 63
7.3.3 Error Tables and Runtime Aspects ................................................. 64
8 Conclusion .......................................................................................... 67
8.1 Results Summary ............................................................................... 67
8.2 Further Improvements ......................................................................... 68
8.3 Conclusion ......................................................................................... 69
A Parameter Optimization
   A.1 Parameter Optimization Plots ............................................ 70
   A.2 1D Parameters ............................................................. 76
   A.3 2D Parameters ............................................................. 76

B Adaptive Linear Filter ......................................................... 77

C Experiment Result Tables ..................................................... 78
   C.1 Result Table Abbreviations .............................................. 78
   C.2 General 1D Profile Experiments ....................................... 79
   C.3 2D Profile Experiments .................................................. 80

Bibliography ............................................................................ 81
List of Figures

1.1 LLASM Simple Pipeline Architecture Overview. ........................................ 13
3.1 56 aligned lip shapes. .............................................................. 18
3.2 The mean shape \( \bar{x} \) with the three first principal components weighted with \( \pm 3\sqrt{\lambda_i} \). 20
3.3 1D profile directions along the inner and outer lip contour and sample profile. .... 20
3.4 Example for misplaced shape points. .................................................. 23
4.1 CUAVE56-L landmark positions. ..................................................... 30
4.2 Typical distribution of localization errors. .......................................... 32
5.1 Aligned shapes using procrustes analysis. .......................................... 36
5.2 Interpolated lip shape ................................................................. 39
5.3 Search windows when using 2D patches. .......................................... 40
5.4 Mean patch \( \bar{p}_k \) with the three first eigenpatches weighted with \( \pm 2\sqrt{\lambda_i} \). .... 41
5.5 Example of a two-dimensional Gabor filter. ....................................... 43
5.6 Gabor patches extraction and PCA model building during training. .......... 44
5.7 Best-fit-distances histogram and example match weight function. ............. 47
5.8 RGB histograms of all mouth ROI regions in the CUAVE56-L data set and example original and equalized mouth ROI patch. ........................................ 49
6.1 Uncontrolled shape drift when using the output shape from the previous frame as initialization. ................................................................. 52
6.2 The ASM is unable to follow the face’s horizontal translation motion. ........ 53
7.1 Comparison of grayscale and color PCA patches. ................................ 59
7.2 Progression of error reduction when combining working improvement approaches with 1D profiles. ......................................................... 61
7.3 Progression of error reduction when combining improvement approaches with 2D patches. ........................................................ ...... 62
7.4 Frame-per-Frame tracking with 2D profiles. ....................................... 63
7.5 Bounded vs. unbounded reuse of output shape of the previous frame. .......... 64
7.6 Translated vs. untranslated reuse of the output shape of the previous frame. .... 65
A.1 Optimization of the number of eigenvectors used with 2D PCA patches. ....... 71
A.2 Cumulative sum of PCA patch eigenvalues. ............................. 71
A.3 Optimization of the parameter 2D patch width. ...................... 72
A.4 Optimization of the parameters pyramid level count and pyramid factor. ............. 72
A.5 Cumulative sum of PDM eigenvalues. ................................. 73
A.6 Optimization of the number of eigenvectors used on different pyramid levels. .... 74
A.7 Optimization of the parameter interpolation factor. ..................... 75
A.8 Optimization of the match weights function $f(d_k)$. .................... 75
List of Tables

4.1 Reference detection quality values achieved by STASM and LLASM. 33

5.1 RGB histogram equalization remapping table. The input ranges of the individual channels given in the table is scaled to the full possible output range $[0..255]$ (uint8). 48

7.1 Localization quality values achieved by different initialization methods. 54
7.2 Localization quality improvement comparison with different alignment methods. 55
7.3 Localization quality comparison of Cartesian and polar shape models. 56
7.4 Localization quality increase when using interpolated shapes. 57
7.5 Localization quality comparison of different 2D patch approaches. 57
7.6 Localization quality with and without eye-distance scale normalization. 58
7.7 Localization quality comparison of color and grayscale 1D profile types. 58
7.8 Localization quality increase when using match weights. 59
7.9 Localization quality increase when varying number of eigenvectors on different pyramid levels. 60
7.10 Localization quality comparison with and without image pre-equalization. 60
7.11 Localization error and runtime comparison between different tracking implementations using 2D patches. 65
7.12 Localization error and runtime comparison between different tracking implementations using 1D profiles. 66
# List of Algorithms

1. The principal component analysis (PCA) ........................................ 18
2. The abstracted ASM search algorithm ........................................ 22
3. Fitting a model shape to a target shape ....................................... 24
4. The ASM search algorithm, including Multi-Resolution Search ........ 26
5. Testing algorithms by partitioning the data set ............................. 31
6. The PCA patches training algorithm ........................................... 42
Chapter 1

Introduction

1.1 Motivation

The visual appearance of the human mouth holds a lot of information about the individual it belongs to. It is not only a distinct part of each person’s looks, the lip shape also serves as mean of expressing our emotions. Moreover, the lips’ motion indicates if the person is talking and even allows conclusions about what is being uttered. Therefore, localizing the exact lip boundaries in an image or video are a demanded, valuable information for various applications with human-computer interaction and in automated surveillance.

The most prominent example is automatic speech recognition (ASR), which is considered an integral part of future human-computer interfaces. Current state-of-the-art audio based ASR systems still lack of robustness to channel and environment noise and are lagging human speech perception by magnitudes [1, 2]. The emerging field of study called audio-visual speech recognition (ASVR) aims to correlate a speech signal with the speaker’s mouth’s visual cues, which is given when a video image of the speaker is available. Humans use this technique unknowingly, especially in difficult acoustic situations, which is demonstrated by the so called McGurk effect [3]: When for example the spoken sound /ba/ is superimposed on a video of a person uttering /ga/, most people will perceive /da/.

It should be noted that it is generally next to impossible to perform automatic speechreading, i.e. lip reading without audio-information, like hearing impaired or deaf people successfully apply [1]. This is due to the increased ambiguity of visual speech features compared to audio, e.g. the mouth position when producing the sounds /f/ and /v/ are practically identical. Human speechreading requires thorough understanding of spoken language but first and foremost also of the situational context and the pragmatic meaning of an utterance - something ASR systems are still far away from achieving.

For being able to extract the mouth’s visual features during speech, it is very desirable that the lips are precisely localized, finding the whole mouth is not enough [4]. The lips are very deformable and vary in shape and color from person to person, what makes them particularly difficult to locate and track accurately and robustly, especially with different lighting situations.

There exist many approaches for achieving this, that will be described in chapter 2. Although this may appear a very central point in audio-visual speech recognition, most current research in
Chapter 1. Introduction

this field focuses on advanced parts of ASVR systems, like the extraction of visual features and the combined audio-visual classification.

To the best of our knowledge, currently there is no solid lip tracking solution available publicly, nor are there databases that provide extensive test data for research on visual feature extraction and the subsequent classification steps. Apparently, the investigations on advanced steps have been conducted with specialized data, like close-up videos of mouths with marked lips.

1.2 Method

In this project, a functional lip shape tracker based on multiple detection methods was implemented. The core part of program is a technique called Active Shape Models (ASM), which is used for the final, accurate lip contour localization. This master thesis focuses on implementing and improving the ASM localization methods. ASMs have been first introduced by Cootes and Taylor [5] and can be applied to many shape localization tasks. This technique was chosen because it has been described as robust and versatile [6].

Prior to the ASM part, a number of preprocessing steps are required in order to obtain more information on what is in the image, which are illustrated in figure 1.1.

In the face detection step, the face regions in the image frame are detected and localized. The algorithm selected for this is using the face detection method proposed by Viola and Jones, and will also be referenced as Viola-Jones face detector [7].

In each found face frame, an eye detector is executed to find both eye positions. The algorithm that provides that uses isophote curvatures and was described by Valenti and Gevers [8]. Next, a smaller part of the face frame containing only the mouth is found, using the eye locations as indicators to the mouth’s position. This mouth-frame is called the region-of-interest (ROI). The method used for that was described in a paper entitled Hough Transform-based Mouth Localization by Fanelli et al. [9]. It is a special application of class-specific Hough forests as described by Gall and Lempitsky [10].

Given the face frame, the eye locations and the mouth ROI, the more accurate lip localization using ASMs is conducted. This process will be elaborated in detail throughout the next chapters.

1.3 Architecture Overview

Our complete lip localization program is called LLASM, which stands for Lip Localization Active Shape Models. As already shown, the pipeline for processing one single frame consists of four main processing parts (figure 1.1).

The first three parts are implemented as one single application in C++, using the OpenCV library, which provides useful tools for image processing. In particular, the Viola-Jones face detector using Haar-feature classifier, is already implemented in OpenCV. The eyes and mouth ROI detector also use many OpenCV functions, but their code comes from other research projects and is still experimental. We used this external code and left it largely unmodified. We assume that the detectors all work as intended, while in particular the eye detector returns incorrect locations at times.
The lip localization part with ASMs, on which this master thesis’ investigations were conducted, was implemented in Matlab for easier data handling and code simplicity, which is helpful when trying out different improvement ideas. The ASM module and the face feature detector program communicate by text files containing the frame and feature locations.

The control of the whole processing pipeline algorithm is also done with Matlab. In other words, for a single frame the pipeline script calls the C++ program first and starts the ASM lip localizer subsequently. This control unit is then later extended to a lip tracking program.

1.4 Layout of this Report

In the next chapter, we will briefly review related works on lip localization. The remaining chapters will cover the investigations made on ASM lip localization. Therefore, chapter 3 will briefly introduce the ASM method in its original form, which was designed for localizing the features of whole faces.

The main contributions of this master thesis then follow. In order to improve localization quality of the ASM-based algorithm, a variety of improvement approaches were investigated. The suitability testing conditions, such as how localization quality was measured, and the data base used for those tests is stated in chapter 4. Chapter 5 then explains the individual algorithm improvement approaches in detail. Chapter 6 describes how the single-frame processing pipeline was extended to a lip tracking program and what needed to be considered when doing so. The numerical results of the approaches described in chapters 5 and 6 as well as their immediate discussion are given in chapter 7.

The final chapter 8 summarizes the findings of the previous chapters, draws conclusions and points out further steps.
Chapter 2

Related Work

2.1 Lip Localization Techniques

A variety of lip localization methods have been described in the literature throughout the last 15 years. Popular approaches base on color and intensity thresholding to segment the lips from the rest of the face [11, 12, 13, 14, 15]. Usually the lips are then located by fitting a shape model around the segmented mouth, where many techniques were investigated. Another popular method is the use of snakes in combination with mouth corner feature detection [16, 17]. Also, shape templates have been used in order to localize the lip contours [18]. Another approach is to classify the areas in an image according to the horizontal and vertical intensity profiles, with special consideration of the different casting of shadows in the mouth area [19].

There are several publications that specifically focus on real-time lip tracking. They often use the same methods as mentioned above, maybe as simplified or speed-up variants. For example [20, 21, 22] use the same color segmentation technique as described above. The color segmentation-based approaches are often lacking robustness to changes of lighting and speakers, but in particular to facial hair. An interesting solution to this was proposed by Petajan et al. [23], where the nostrils’ openings were used to determine the approximate mouth location and to estimate the facial hair. A simpler approach was proposed by Yang et al., which only searched for six characteristic points on the lip with characteristic corner features [24].

2.2 Lip Localization with Active Shape Models

This master thesis is not the first attempt of applying Active Shape Models to the problem of finding lips, early investigations are documented by Luettin et al. [25] and Matthews et al. [26]. However, due to the lack of implementation and testing details in the former publication and with no program code or test data available whatsoever, we were unable to recreate those results and focus on further AVSR processing steps. In the latter publication, the research focus is not on robust and versatile lip localization ASMs but on suitability of PCA-space representations of the lips as features for AVSR. On that account, the models were trained with excerpts from the same sequences it was later used to detect the lips in. In our project, speaker-independent models are aimed for. In this context, it may not appear surprising that our initial localization implementation
lacked of versatility, accuracy and robustness, so we decided to investigate more on this subject.

The two aforementioned works apply the original ASM formulation by Cootes and Taylor [5] with only one major change: instead of the original profile model of local appearance, which consists of mean and covariance matrix, a PCA based gray-level distribution model (GLDM) was introduced. On the other hand Matthews et al. also investigated the use of an extension of ASM, called active appearance model (AAM). The AAM’s parameter vectors provide more suitable visual speech features, but it is not stated how well or robust the localization was working with regard to illumination variations and different speakers.

In a more recent paper Jang et al. propose the use of Gaussian mixture models (GMM) as a replacement for the GLDM [27]. Although the overall detection quality only improves slightly, the placement of the inner lip contour was significantly improved by this mean.
Chapter 3

Active Shape Models

The term Active Shape Model (ASM) is associated with a set of techniques used to detect the contours of a certain object in an image. We place the object’s mean shape over the image and deform it to fit the contours of the underlying object, therefore the name ‘active’. The basic technique was introduced by Cootes & Taylor in 1995 and has been applied and adapted in numerous tasks ever since [5]. A good reference is Locating Facial Features with Active Shape Models by Milborrow [28], which explains the ASM algorithms in a down-to-earth way and is recommended to read as a practical introduction to the topic.

This chapter reviews the ASM methods proposed by Cootes and Milborrow which are relevant for this project. Both reports describe how to locate facial features, in contrast to that, this chapter will already apply the concepts to lip shapes.

3.1 Overview

A shape, in the context of Active Shape Models, is nothing else than a set of connected two-dimensional points (or point cloud). The connections however have no stringent influence on the model’s deformation behaviour. In the standard ASM, the number of points always stays the same, and each point should always correspond to the same feature on the object’s image. The shape’s points can be regarded as distinct feature points of the object, e.g., the mouth corners are good feature points in a lip shape.

Using Active Shape Models to locate object boundaries is a machine learning task and it requires a training and a search algorithm. The model built during training and used during search consists of two main components:

1. The shape model or point distribution model (PDM), that describes the mean shape and the variance of the shapes observed during training. This is done by analyzing the points’ relative position to each other in all the samples. In other words, it is a model that is able to produce valid mouth shapes, i.e., like the ones seen during training.

2. The profile model, which describes each shape point’s typical neighborhood appearance when placed on the object boundary (also called local structure model).
The following sections describe the properties of those components and how they are built and used in this work, structured according to the algorithms training and search.

### 3.2 ASM Training

To train the Active Shape Model, a set of images with landmarked feature points is required, i.e., each sample consists of an image, containing the object, and an annotation, which features a list of pixel coordinates marking the shape’s feature points in the image. The training set actually used in this work is described in section 4.1.

#### 3.2.1 The Shape Model

The shape model is built by performing a principal component analysis (PCA)\(^1\) on shape-point coordinates given in the training set. In order for PCA to work, the sample shapes must be aligned. This is because the sample object’s shape may be in completely different locations in the image, rotated and in different scales. Their points must be transformed into a common coordinate system to calculate an average shape and to perform the analysis of the shape changes.

One method used by Cootes and Milborrow is **Procrustes analysis**. The output of this analysis is a 3x3 transformation matrix that aligns a given shape to a reference shape. This transformation minimizes the sum of distances between two corresponding shape points using a least-squares approach. A more detailed description of this method can be found in Appendix C of Cootes’ report [29], where the affine and similarity cases are considered.

Given this, we can first align all the lip shapes, i.e., point clouds, which will result in an aligned data set as depicted in figure 3.1.

After this, the sample’s data need to be vectorized, since PCA can only analyze data vectors, not matrices. This can be done in various ways, but this choice has no influence on the output. Here is how it was done in this project, also introducing the notation:

The data set consists of 56 samples \(x_k, k \in [1, \ldots, 56]\), each one holds a shape consisting of 20 points \(p_{k,i} = (x_{k,i}, y_{k,i}), i \in [1, \ldots, 20]\), with \(x_i\) and \(y_i\) being the point’s pixel coordinates. The coordinate values of each sample are then arranged into one vector as\(^2\): \(\vec{x}_k = (x_{k,1}, x_{k,2}, \ldots, x_{k,20}, y_{k,1}, \ldots, y_{k,20})^T\).

Once the data is vectorized, the PCA can finally be performed. There exist multiple methods for calculating this, the used one will be described here. Algorithm 1 shows the steps conducted during the calculation.

---

1. PCA is also called Karhunen-Loéve or Hotelling transform.
2. In MATLAB this can be easily done by using colons as index of a sample matrix: \(S_k(:,\cdot)\)
Figure 3.1: 56 aligned lip shapes. Each color represents one feature point.

**Algorithm 1**: The principal component analysis (PCA)

**Input**: n samples $\mathbf{x}_k$

**Output**: $\bar{x}, \Phi, \lambda$

\[
\bar{x} = \frac{1}{n} \sum_{k=1}^{n} \mathbf{x}_k ;
\]

\[
S = \frac{1}{n-1} \sum_{k=1}^{n} (\mathbf{x}_k - \bar{x})(\mathbf{x}_k - \bar{x})^T ;
\]

\[
\lambda = \text{EigenValues}(S);
\]

\[
\Phi = \text{EigenVectors}(S);
\]

or in MATLAB code:

```matlab
% given n vectors x_k as: x(k,vector-index)
xmean = mean(x,1);
S = cov(x-repmat(xmean,1,n));
[phi lambda] = eig(S);
```

$\bar{x}$ is the mean shape and $S$ is the covariance matrix of the sample dimensions. The PCA outputs $\bar{x}$, $\Phi$ and $\lambda$ are used in the following equation:

$$\hat{x} = \bar{x} + \Phi \cdot b \quad (3.1)$$
Here, $\hat{x}$ is a new instance of shape, a mixture of samples we have seen during the training. The equation can be understood as adding shape-characteristic vectors ($\Phi$) to the mean shape $\bar{x}$, where each vector in $\Phi$ is weighted by an element of $b = (b_1, b_2, ..., b_n)^T$. In simple terms, by specifying $b$, we obtain a valid shape.

However, $b$ cannot be chosen freely, its elements need to lie within certain bounds for the equation to produce a plausible shape. Since $\Phi$ contains eigenvectors of the variance, by bounding the values of $b$ to

$$|b_i| < \sqrt{\lambda_i}$$

(3.2)

the model will be able to reproduce all the samples seen during training. If we want the model to allow even a bit more extreme shapes, this limit can be increased, e.g. up to $3\sqrt{\lambda_i}$ [29]. This is needed if not all of the variance was seen during training, which is the usual case.

By sorting the eigenvectors in matrix $\Phi$ according to the size of the corresponding eigenvalue $\lambda_i$, the most principal components can be found, i.e., the larger the corresponding eigenvalue, the larger amount of variation that is covered by its eigenvector. It is therefore expedient to only take $k$ eigenvectors with large eigenvalues into account, since we want to keep important aspects of the training shapes, but ignore noise. This leads to a dimensionality reduction of $b$, the feature vectors’ representation in the PCA space. The amount of variance covered by the subset of eigenvectors can be calculated as

$$\mu(k) = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{n} \lambda_i}$$

(3.3)

In this project, the optimum number of eigenvectors to retain was determined by empirical testing. Alternatively, this number can be determined by a certain percentage of the total variance $\mu(k)$ that shall be covered.

### 3.2.2 The Profile Model

The idea of the profile model is to determine the characteristic neighborhood of a shape feature point in an image. The search algorithm uses this model to find the best corresponding point in its vicinity, i.e., it compares how much a candidate point’s neighborhood resembles the one of the model, and picks the most similar.

There exist multiple approaches to do this, the most basic but effective one described by Cootes uses 1D-profiles (also called whiskers) [5]. For modeling each feature point’s neighborhood, only a one-dimensional profile is extracted from the image around each point. The axis of this profile runs through the feature point and is orthogonal to the shape’s contour, i.e., it is orthogonal to the connecting line between the two adjacent shape points. This is due to the assumption that the shape line runs along an edge in the image, and that the shape grows or shrinks in this direction. Figure 3.3 illustrates how the whiskers are positioned on a lip shape.

The first step in 1-D profile modeling is to extract all the profiles, i.e., from each point and from each sample ($N_{\text{Samples}} \cdot N_{\text{Points}}$). Milborrow and Cootes only use grayscale images, which makes the profiles easier to process. To lessen the effect of overall lightness and contrast due to lighting changes, the profiles are preprocessed: first, the gradient $p'$ of the profile is calculated. This is done
Figure 3.2: The mean shape $\bar{x}$ with the three first principal components weighted with $\pm 3\sqrt{\lambda_i}$. 

Figure 3.3: Left: 1D-profile directions along the inner and outer lip contour. Right: a sample profile.
by calculating the difference of neighboring pixels in the profile:

\[ p = [g_1, g_2, g_3, \ldots g_{Np}] \rightarrow p' = [(g_1 - g_2), (g_2 - g_3), (g_3 - g_4), \ldots (g_{Np-1} - g_{Np})] \quad (3.4) \]

where \( Np \) is the number of values or length of each profile. There are \( N_{\text{Samples}} \cdot N_{\text{Points}} \) profiles \( p \) in total, the indices are omitted for readability.

Second, the gradient profile is normalized, such that the sum of all vector elements is one:

\[ p' = [g'_1, g'_2, g'_3, \ldots g'_{Np}] \rightarrow \hat{p}' = \frac{p'}{\sum_{i=1}^{Np} |g'_i|} \quad (3.5) \]

After this preprocessing the actual model is built: Cootes assumes that the profiles are approximately distributed as a multivariate Gaussian, and therefore can be described by their mean \( \bar{p} \) and covariance matrix \( S \). Thus, for each point \( k \), we will get:

\[ \bar{p}_k = \frac{1}{N_{\text{Samples}}} \sum_{p' \text{ of point } k} \hat{p}' \quad (3.6) \]

\[ S_k = \frac{1}{1 - N_{\text{Samples}}} \sum_{p' \text{ of point } k} (\hat{p}' - \bar{p}_k)(\hat{p}' - \bar{p}_k)^T \quad (3.7) \]

The difference between two profiles \( x \) and \( \bar{p}_k \) during search can then be calculated using the Mahalanobis distance:

\[ \text{Distance} = \sqrt{(x - \bar{p}_k)^T \cdot S_k \cdot (x - \bar{p}_k)} \quad (3.8) \]
3.3 ASM Search

As mentioned in the section above, preliminary to the ASM search algorithm, a region-of-interest (ROI) detector must be run in order to achieve a good initialization. For faces, this may be a Viola-Jones face detector, for lips a more constrictive ROI detector will be used (section 5.2). For this explanatory chapter, we will assume the use of a Viola-Jones frame. Given an image and a face frame, the algorithm performs in the following way, using the three components described before:

Algorithm 2: The abstracted ASM search algorithm

<table>
<thead>
<tr>
<th>Input:</th>
<th>Image, ROI, Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>Shape $Y$</td>
</tr>
<tr>
<td>$Y = \text{initializeShape(ROI, Initialization Method)}$;</td>
<td></td>
</tr>
<tr>
<td>while not converged do</td>
<td></td>
</tr>
<tr>
<td>\hspace{1em} foreach Point $Y_i$ do</td>
<td></td>
</tr>
<tr>
<td>\hspace{2em} $X_i = \text{findBetterFit}(Y_i, \text{Image, Profile Model})$;</td>
<td></td>
</tr>
<tr>
<td>\hspace{2em} $\hat{X} = \text{findFittingShape}(X, \text{Shape Model})$;</td>
<td></td>
</tr>
<tr>
<td>\hspace{2em} converged = checkConvergence($Y, \hat{X}$);</td>
<td></td>
</tr>
<tr>
<td>\hspace{2em} $Y = \hat{X}$;</td>
<td></td>
</tr>
<tr>
<td>return $Y$;</td>
<td></td>
</tr>
</tbody>
</table>

This algorithm is a simplified version that illustrates the basic steps. The remainder of this section will illustrate the used pseudo-code functions and give a more in detailed look on why and how this works.

initializeShape(ROI, initialization method)

As mentioned in the introduction of this chapter, at the beginning of the search algorithm, a mean shape is initialized at a location where the shape is assumed to be. This can be done in various ways, the original one used by Cootes and Milborrow scales the mean shape relative to the ROI given in the image. The details of this process are described in the following subsection 3.13.

findBetterFit(Shape Point, Image, Profile Model)

Within a given search window, the feature point is moved to the location that best fits the model neighborhood.

In the basic implementation by Cootes and Milborrow, the search window consists of a number of points along the whisker. This is done as follows:

1. First, the whisker direction is calculated by taking the orthogonal vector to the connection line of the adjacent points, then this vector is normalized.
2. Next, a profile $a_k$ is extracted along this vector, which is longer than the model profile $\bar{p}_k$, usually twice as long. The double length allows the point to move in one iteration exactly as far as the model whisker reaches.

3. The model profile is shifted over the extracted profile, where the Mahalanobis distance (equation 3.8) is calculated at each position.

4. The point is moved to the location with the lowest Mahalanobis distance, i.e., the best fit between the model $\bar{p}_k$ and the profile $a_k$.

This is done for each point, distorting the shape. Although the points’ neighborhoods are now at their best fit within their search windows, a lot of the points placements are usually not correct yet. This occurs especially with feature points at weak edges, may it be in the search image or in the training data, but also at strong edges, if the profile extracted from the search image runs across multiple edges.

Figure 3.4 illustrates this problem: the mouth corners are badly placed because of the lack of features in this area. The left and right outer corners therefore even stick to the more contrast rich edge of the moustache. One point of the inner lip contour is clearly placed wrongly, because the edge between the teeth and the dark inside of the mouth formed a stronger contrast than the edge between lip and teeth. For all those cases, the next function is used to correct those shape deformations.

The shape model is used to generate a shape instance, that, when correctly placed over the original, minimizes the sum of all point distances. In other words, under the assumption that only few points are misplaced, the resulting shape will feature those points in their right places. As an example, the heavily misplaced point on inner lip contour in figure 3.4 will most likely be moved, since the model lip shapes usually are symmetric - only smaller asymmetries can be reproduced.
findFittingShape(Shape, Shape Model)

In this function a shape instance of the PDM is found that best fits the target shape provided by
the previous step. The target shape here is the one obtained from the local fitting process. The
model shape is produced by equation 5.8, and then translated, scaled and rotated onto the image.
The question therefore is to find the shape parameter vector $b$ and a transformation matrix $T$,
which minimizes the distance between the new and the model shape. Expressed in equations, this
means:

$$\arg\min_{b,T} \|Y - \hat{X}\|$$

(3.9)

where $Y$ is the target shape and $\hat{X}$ is the model shape, which is formed as:

$$\hat{X} = T(\hat{x}, T) = T(\hat{x} + \Phi b, T)$$

(3.10)

$$T(\hat{x}, T): \quad \hat{X}_i = T \cdot \begin{pmatrix} \hat{x}_i \\ 1 \end{pmatrix} = \begin{pmatrix} s \cos \theta & s \sin \theta & t_x \\ -s \sin \theta & s \cos \theta & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \hat{x}_{i,x} \\ \hat{x}_{i,y} \\ 1 \end{pmatrix}$$

(3.11)

Equation 3.11 illustrates the similarity transform for each single shape point $\hat{x}_i$, where the
transformation matrix $T$ is constructed with the parameters translation vector $t$, scale $s$ and
rotation angle $\theta$.

Cootes & Taylor propose the following iterative algorithm for finding the parameters:

**Algorithm 3: Fitting a model shape to a target shape**

**Input:** Target Shape $Y$, Shape Model

**Output:** Model Shape Instance $\hat{Y}$

Set $b=0$;

**while** not converged **do**

\[
\begin{align*}
\hat{x} &= \hat{x} + \Phi b; \\
T &= \text{Procrustes}(\hat{x} + \Phi b, Y); \\
\text{Project } Y \text{ to the model coordinate frame: } y &= T(Y, T^{-1}); \\
\text{Update } b &= \Phi^T \cdot (y - \hat{x}); \\
\text{Apply constraints to } b \text{ (e.g. } |b_i| < 3\sqrt{\lambda_i}); \\
\text{Check convergence by evaluating the match error } e &= \sum_{k=1}^{n_{pts}} \|y_k - (\hat{x} + \Phi b)_k\|;
\end{align*}
\]

$\hat{Y} = T(\hat{x} + \Phi, T);$
the similarity or affine transformation matrix $T$, that best matches the first shape to the other.

**checkConvergence(Shape, Shape)**

At the end of each iteration loop, a convergence check is done. This can be achieved by simply calculating the sum of distances between each of the corresponding shape points and checking whether the difference is larger than a certain empirically found threshold. However, Cootes proposes to use boolean flags “found” with each point, which are set to `true` in the profile fitting phase if the point has been moved less than a given number of pixels. At the convergence check function, the portion of “found” points is evaluated, and the solution is declared as converged if it is large enough.

### 3.3.1 The Initialization Method

The shape initialization is critical for the search algorithm’s convergence and therefore needs special consideration. Since Milborrow is focusing on locating facial features, his initialization method is started from a Viola-Jones or Rowley frame, i.e., a frame returned from a Viola-Jones or Rowley face detector. For finding a good initialization position the average position $\bar{c}_i$ within the face frame is calculated for each point $i$:

$$\bar{c}_i = \left(\bar{x}_i, \bar{y}_i\right) \tag{3.12}$$

$$\bar{x}_i = \frac{1}{N_{\text{Samples}}} \sum_{k=1}^{N_{\text{Samples}}} \left( x_{ki} - F_{kx1} \right) \frac{F_k \text{width}}{F_k \text{width}}$$

$$\bar{y}_i = \frac{1}{N_{\text{Samples}}} \sum_{k=1}^{N_{\text{Samples}}} \left( y_{ki} - F_{ky1} \right) \frac{F_k \text{height}}{F_k \text{height}}$$

Where $F_k$ is the face frame, with its width and height, and $(F_{kx1}, F_{ky1})^T$ being the top left point of the frame$^3$. $x_{ik}$ and $y_{ik}$ are the pixel coordinates of point $i$ in sample $k$. At the start of the search, each point $Y_{s_i}$ of the shape will reversely be initialized within the face frame $F$:

$$Y_{s_i} = \left( \frac{(\bar{x}_i + F_{x1}) \cdot F_{\text{width}}}{(\bar{y}_i + F_{y1}) \cdot F_{\text{height}}} \right) \tag{3.13}$$

### 3.3.2 Pyramid Level Search

As a last thing that needs to be introduced in the context of ASM search, Cootes and Taylor propose pyramid level search, i.e., the sequential search on multiple image resolution levels [29]. This measure is assumed to increase efficiency and robustness of the search algorithm. Practically, search algorithm 4 is run on every resolution level until convergence, starting from the coarsest resolution and finishing with the highest. The shape is rescaled to the next resolution level after reaching convergence. The training also needs to be adapted: the shape model is the same across all pyramids, but a separate profile model is needed for each resolution level.

---

$^3$Assuming the use of the standard image coordinate system with origin in the top left corner.
Below the adapted pyramidal search algorithm:

**Algorithm 4:** The ASM search algorithm, including Multi-Resolution Search

**Input:** Image, ROI, *Model*  
**Output:** Shape *Y*

Build Gaussian image pyramid (lower resolution images);  
*Y* = initializeShape(ROI, *Initialization Method*);

**foreach** Pyramid Level **do**

- \( \hat{Y} = \text{Scale down } Y; \)
-Scale down Image;

  **while** not converged **do**

    **foreach** Point \( \hat{Y}_1 \) **do**

    - \( X_1 = \text{findBetterFit}(\hat{Y}, \text{Image, Profile Model}); \)
    - \( \hat{X} = \text{findFittingShape}(X, \text{Shape Model}); \)
    - converged = checkConvergence(\( \hat{Y}, \hat{X} \));
    - \( \hat{Y} = \hat{X}; \)

- \( Y = \text{Scale back up } \hat{Y}; \)
Chapter 4

Data Acquisition and Model Evaluation

4.1 Training Data

4.1.1 Requirements

Building the Active Shape Models requires training data in the form of images with annotated feature points. This means that we need a number of samples, where each consists of an image and a set of pixel point coordinates, which form the shape we look for. Typically, for each image, there is an accompanying text file containing those coordinates.

The training data should contain all the variation expected during search. Since the focus of this Master Thesis lies on detecting lip shapes for audio-visual speech recognition, the training data has to fulfill the following requirements:

1. images or movies of frontal faces
2. the mouth shape points must be manually landmarked in each frame, and contain detailed lip feature points (at least inner and outer contour)
3. subjects must be talking
4. varying speakers’ identity
5. image resolution must be satisfyingly high

Unfortunately, no database that fulfilled all of the above requirements was found.

4.1.2 Existing Databases

There are multiple databases that have been considered for this project:

- BioID [30]
  The BioID data set consist of 1521 384x286 pixel monochrome images of 23 different subjects. The faces are manually landmarked with 20 points. The images are frontal views of various
people taken inside an office interior. It seems that the camera has been moved around to
the respective subject’s desk, thus the lighting, angle and distance to the camera vary a
lot. Also, the subjects show different facial expressions and sometimes are talking. The low
number of landmarked points and the relatively low resolution restrained the author from
using this database for training ASMs. Nevertheless, this database can still be used as test
set to measure an algorithm’s detection quality.

- **CUAVE[31]**
  CUAVE stands for Clemson University Audio-Visual Experiments. They provide a database
  which consists of movies of 36 individuals, uttering sequences of numbers. Speakers either
  stand still or move while uttering continuous or contiguous numbers from zero to nine, con-
  tinuously or with short pauses. There are frontal view and side view sequences in each video.
  Furthermore there are 20 videos showing two speakers in the same frame, speaking alter-
  nately. The individuals have different accents, skin tone, facial hair and glasses. Additionally,
  each video includes a transcript with timestamps for each uttered number. The videos are
  recorded in a controlled lighting environment in front of a green screen, the lighting however
  appears to change slightly between the shots. The video resolution is 720x480, encoded as
  5Mbps MPEG-2 with 44 kHz audio.
  Unfortunately, this database does not satisfy the requirement of landmarked feature points.

- **Talking Face Video[32]**
  The “Talking Face Video” is a database that contains 5000 color frames of a single person
  with 68 semi-automatically landmarked face feature points. It was created by the Face and
  Gesture Recognition Working group (FGnet) and is sometimes also referred to as “Franck
  DB”, because the files all titled with this name. The video was recorded with a blue screen
  in the back. The automatic landmarking, which induces minor positioning errors, and the
  fact that only one speaker is available makes this database only usable for detection quality
  measurement. In this project, this database was used for tracking method evaluation.

- **XM2VTS[33]**
  The XM2VTS frontal image sets contain 2360 720x576 color images of 295 subjects. The
  pose and lighting is uniform, with a flat background. The faces are manually landmarked
  with 68 points, containing a sufficiently fine annotation of lip feature points. Another set of
  the XM2VTS database contains videos of talking people, similar to the CUAVE samples, i.e.,
  the videos do not come with annotated shape points neither. The XM2VTS data must be
  purchased and was not available for this project.

### 4.1.3 The CUAVE56-L Database

To get a suitable data set to fulfill the requirements mentioned above, a subset of the CUAVE
database was taken:

1. To keep the option of using this database for research on multilinear PCA (isolate the effects
   of different identity and different visemes), the idea is to have the same phonemes and visemes
spoken by different people, what gives us the corresponding visemes\(^1\). Using the annotated timestamps, a number of frames were extracted from the video at the time the subject voicing a certain phoneme. Multiple frames were extracted to catch the viseme at its strongest occurrence. Due to the limited set of distinct visemes in the English digits zero to nine, seven visemes were extracted per subject. The choice of these visemes is rather highhanded and does not at all represent a linguistically well-founded set - this was relinquished due to the already limited set of visemes and the limited time available. The following seven visemes were extracted from eight people, resulting in 56 sample frames:

- a/e - eight - /æt/
- f - four - /f\textipa{3}\textipa{1}/
- i - three - /\textipa{0}\textipa{i}/
- o - four - /f\textipa{3}\textipa{1}/
- s - six - /s\textipa{k}\textipa{s}/
- th - three - /\textipa{0}\textipa{i}\textipa{e}/
- u/w - two - /\textipa{2}\textipa{n}/

2. Each of the 56 viseme-frames was then manually landmarked using a suitable program, i.e., a text file containing the shape feature point coordinates was created for each sample. In each sample, 23 points were landmarked:

As 4.1 shows,

- Points 1-12 form the outer lip contour (12),
- Points 13-20 form the inner lip contour (8),
- Point 21 is the approximate mouth center,
- Points 22 and 23 are the eye pupil centers.

Note that the 20 lip shape points are the same that are landmarked in the XM2VTS (68 face shape) database, except that the two inner mouth corners were added (points 13 & 17).

### 4.2 Model Evaluation

#### 4.2.1 Partitioning the Data Set

In order to test how well an ASM variant performs, if a certain modification in the training or search algorithm gives better or worse results, we need to partition the data base into training and testing data. The model is then produced by using that training data subset, and the outcome of the search algorithm is compared to the ground-truth annotation of the testing data subset. We do this by using the average error and the pseudo-standard deviation, described in subsection 4.2.3.

\(^1\)A viseme describes the particular facial and oral positions and movements when forming a certain sound (phoneme).
The CUAVE56-L database was used for determining which approaches work well and which do not. Naturally, it is desirable that the test and training sets are both as large as possible, but the small number of samples in this database lead us to set the priority on a large training set - for obtaining a good model in the first place. However, as Milborrow [28] pointed out, it is very important that no person appears in both the training and the testing subset. Since the CUAVE56-L data set contains 8 persons with 7 visemes each, it is manifest to partition the data by taking 7 persons as training set and 1 person into the test set (leave-one-out cross-validation). Each person’s samples can be taken as test set once, so we can do this test on 8 different database partitions. Algorithm Algorithm 5 is a pseudo-code representation that illustrates how this is done.
and summarizes this section.

**Algorithm 5**: Testing algorithms by partitioning the data set

**Input**: Algorithm variation $\mathcal{A}$, CUAVE56-L DB: 8 people, 7 visemes each

**Output**: Localization quality using $\mathcal{A}$: Average error $\mu$, Pseudo-standard deviation $std$

Split the database into 8 parts:

$\text{part}_i = \{\text{trainingset}_i, \text{testset}_i\}$ where

$\text{trainingset}_i = 49$ samples not showing person $i$

$\text{testset}_i = 7$ samples showing person $i$

foreach $\text{part}_i$ do

Build model with trainingset $i$ (ASM Training with $\mathcal{A}$)

foreach sample $k \in \text{testset}_i$ do

Locate the lips in sample $k$ (ASM Search with $\mathcal{A}$)

Calculate $m_{e17}$ between the result and the ground-truth shape of sample $k$

end

end

Calculate $\mu$ and $std$ of all the $m_{e17}$

As indicated, the implicitly used measures $m_{e17}$, $\mu$ and $std$ will be elaborated in the following subsections.

**4.2.2 Error Measurement - $m_{e17}$**

To measure how well a search algorithm has aligned the shape to the ground-truth feature points, the $m_{e17}$ measure was introduced by Cristinacce[34]. The $m_{e17}$ error is calculated as follows:

1. Calculate the mean euclidean distance between each pair of corresponding points in the candidate shape (algorithm output) and the ground truth shape (manually annotated landmarks).

2. Divide this number by the eye distance, that is also taken from the annotated landmarks (Point 22 & 23 in the CUAVE56-L database).

In other words, this is mean pixel error of each point normalized by the subject’s eye distance - to compensate for different face sizes. However, in the case of the CUAVE56-L database eye distances always stay approximately the same (max $\pm 15\%$).

**4.2.3 Overall Quality Measure of Multiple Samples**

To compare the performance of the investigated algorithms, the output quality will be quantified by by taking two measures: the mean and a pseudo-standard deviation. Before this, the $m_{e17}$ distance of each test sample is calculated, i.e., when using the CUAVE56-L database there are 56 such values. Then, the following two measures are calculated:
Chapter 4. Data Acquisition and Model Evaluation

32

Figure 4.2: Typical distribution of localization errors. $\mu = 5.11$, $std = 2.68$

- The mean of all the errors $\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$, also called the average error.

- A pseudo-standard deviation, referred to as $std$ - like the Matlab function used to calculate it. This measure is then defined as:

\[
std = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^2}
\]

where $x_i$ stands for the normalized error $me_{sample_i, groundtruth_i})$. This does not represent the standard deviation of the errors, since their distribution is clearly not Gaussian. This can be seen in figure 4.2, which shows a typical error distribution. Nevertheless it still represents the data’s variance and therefore can be used for comparing different error results.

Throughout this report, these two key values are given as percentages, since they are typically in the range of 0 to 0.1.

4.3 Reference Values - STASM & LLASM

As part of his Master Thesis on ASMs for face feature localization, Stephen Milborrow implemented an ASM search algorithm in C++. This software is called STASM (Steve’s ASM) and its code is publicly available under the GNU General Public License. It is a library that can be used well for detecting the boundaries of a whole face, i.e. exactly what it was designed to do. However, its performance localizing the precise lip borders is rather poor - which was also pointed out in the Milborrow’s report.
The STASM library is highly complex and designed specifically for finding features of the whole face, so it can not be easily modified to locate lips - therefore the investigations of this master thesis were conducted using our own implementation in Matlab. To not confuse the two implementations in this report, our Matlab version is called LLASM (Lip Localization ASM). All the modifications described later on were implemented and tested on LLASM.

Nevertheless the results produced with STASM were also used for this work to serve as a reference. With the implementation of the algorithm described in chapter 3 (Viola-Jones frame initialization, grayscale 1D Profiles, 20 feature points per shape), the algorithm scored a mean error of 9.3% and an \textit{std} of 7.1%. In our re-programming of this algorithm in LLASM, very similar numbers were achieved. When using 2D patches (15x15), described in Milborrow’s report and also implemented in STASM, those numbers were reduced to 9% and 7%. Table 4.1 shows the exact reference values.

<table>
<thead>
<tr>
<th>Code</th>
<th>Implementation Description</th>
<th>average error $\mu$</th>
<th>pseudo-standard deviation $\text{std}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>REF0a</td>
<td>Milborrow 1D - VJ</td>
<td>9.266</td>
<td>7.088</td>
</tr>
<tr>
<td>REF0b</td>
<td>Milborrow 2D - VJ</td>
<td>9.040</td>
<td>7.018</td>
</tr>
<tr>
<td>REF1</td>
<td>LLASM 1D - VJ</td>
<td>9.073</td>
<td>3.776</td>
</tr>
</tbody>
</table>

Table 4.1: Reference detection quality values achieved by STASM and LLASM.

There is a large number of additional parameters that needed to be set in STASM which will not be mentioned here since their choice lacks of relevance. For more details see the extended experiment table in Appendix C. Note that the STASM reference values may actually be slightly lowered by re-optimizing all the possible parameters and techniques in STASM. Nevertheless they serve well enough as starting values, since in most samples the localization was satisfactory.

The above table also shows a noticeable difference in the pseudo-standard deviation, it is almost cut in half with LLASM. This is probably caused by the stacking of shape models in STASM, which has not been used in LLASM. Stacking ASMs basically means to run the complete search algorithm multiple times, using the output shape of the previous run as start shape for the following one. The problem then occurs when the Viola-Jones frame initialization is very improper, e.g. placed over the nose or the chin, which usually leads to a completely wrong final placement of the ASM lip shape. The cascade of ASMs amplifies this effect, because the shape is allowed to deform and diverge more - due to the larger number of possible iterations. On the other hand, if the initial shape is placed correctly, the cascade improves the precision [28].
Chapter 5

Model Variations & Parameters

5.1 Overview and Variation Classification

Since the Active Shape Models as described in chapter 3 (according to Cootes[5] and Milborrow[28]) did not seem to give satisfying results, a large number of model variations have been considered and tested. The said results with the standard ASM implementation were listed in section 4.3.

In order to improve performance, we have come up with numerous approaches that are listed in this chapter. The algorithm variations were classified into four categories:

- **Initialization Variations**
  Variations that change the way the start shape is generated fall into this category.

- **Shape Model Variations**
  When the way the shape model is structured, trained or used during search is modified, the approach is classified as shape model variation.

- **Profile Model Variations**
  Approaches that strongly modify or replace parts of the model of the local appearance (as described in subsection 3.2.1) are listed in this category.

- **Algorithm Variations**
  If none of the above apply, the change of variation belongs to this class. These are mainly functional changes in the search algorithm.

This chapter lists and elaborates these variations that were assumed to improve the detection quality and precision.

The variation approaches were tested by using the methods stated in chapter 4. The results of those experiments and their discussion follows in section 7.1 of the results chapter.

Varying Parameters like the number of PDM eigenvectors used or the length of 1D-whisker profiles are minor variations of the algorithms - the functionality of a method itself does not change. Their optimization is of course important to lower the localization errors. Considerations regarding this task and a brief discussion of important parameters will be given in the last section of this chapter.
5.2 Initialization Variations

In the reference implementation, the shape was initialized using the relative mean position within a Viola-Jones or Rowley face detector frame, as described in subsection 3.3.1. The results of the reference tests clearly showed that the initialization using this technique is not satisfyingly precise, but also very crucial to the overall detection quality rating. Therefore, as already indicated in the introduction chapter, other possibilities were considered to make the Region-of-Interest frame (ROI) smaller and the initialization more accurate. The methods for doing so have already been named in the introduction’s section 1.2. The two additional detectors provide us with the following total information to be used for a good lip shape initialization:

- Viola-Jones face detector frame
- Eye Positions
- Precise mouth ROI frame (mROI)

With this data available, there exist multiple alternative methods to the Viola-Jones frame mean position for the initialization itself:

- Mouth ROI Stretching
  The average mouth shape \( \bar{x} \) is stretched over the mouth ROI frame, or a certain percentage of the ROI’s width and height.

- Mouth ROI Mean Position
  We use the same approach as before (subsection 3.3.1), but instead of the Viola-Jones frame, the ROI frame is used.

- Eye Distance Scaling, Mouth ROI Positioning
  Since the detected mouth ROI frames seem to vary strongly and randomly regarding the size, but not their center, it may be advantageous to only use the central point of the frame. The scale of the lip would be determined by the average width and height of the mouth as a ratio of the eye distance.

- Viola-Jones Frame Scaling, Mouth ROI Positioning
  The same approach as the one above, but using the Viola-Jones frame height and width as indicators of how wide and high the lips should be. Again, the mouth ROI’s center point is used to position the scaled shape.

- Mean Position using Eye-Coordinate System
  Each mouth shape point’s position can be expressed relative to the the eyes’ positions. The origin of this coordinate system is the middle point of the line between the two eye pupils. The first unit vector of this coordinate system is the vector from the mid-eye point to the right eye. The second unit vector is obtained by rotating the first one by 90°.
5.3 Shape Model Variations

5.3.1 Alternative Shape Sample Alignment

To align the sample shapes for training, Cootes proposed to use Procrustes analysis, which was already mentioned in section 3.2.1. This variant of the Procrustes analysis may introduce nonlinearities into the shape model, i.e., even if certain points were only moving on straight lines between samples, the alignment process may make this movement nonlinear. This can be illustrated when considering two lip shapes: a laughing mouth and a mouth forming a “u” vowel, as depicted in figure 5.1. Their Procrustes analysis alignment to the mean shape is not showing the actual, natural movement of the lip corners (dashed blue lines). In other words, the lip shape’s real-life deformation is not correctly captured and represented when using this alignment. Although it is debatable if this is actually necessary, the following results achieved by modifying the sample alignment method support this supposition.

This subsection will now describe four simple alternative alignment methods. The experimental results of all these approaches are covered in table 7.2 (chapter 7).

Since these approaches are new and their idea cannot be captured accurately by one obvious term each, they were given reference names. The names are given in brackets after each paragraph title (e.g. ’No Scaling’) and are also used in the result tables.

Removing Sample Rescaling (No Scaling)

As a first step towards more appropriate alignment, removing the Procrustes analysis’ scale component was assumed to be advantageous. Since the faces in the CUAVE56-L data set are all about the same size, this could be done by simply removing the scaling of the shape during the alignment process. Practically, this was done by separating the transformation matrix’ rotation and scaling
parts using singular-value decomposition. We speak about a 2x2 transformation matrix $R$ here, which only forms the upper left corner of the 3x3 transformation matrix $T$ described in section 3.3.

$$R = U\Sigma V^*$$  \hspace{1cm} (5.1)

In equation 5.1, showing the SVD of $R$, the matrices $U$ and $V$ represent two size-preserving rotations, which is given by their required orthonormality. The diagonal matrix $\Sigma$ holds the factors of two-dimensional scaling\(^1\) conducted between those rotations. By setting the scaling matrix $\Sigma$ to the identity $I_2$, the shapes’ size remains unchanged. So the rotation matrix used is calculated as shown in equation 5.3.

$$\hat{R}_{NoScale} = UI_2V^*$$  \hspace{1cm} (5.2)

Adding Eye Distance Scaling (Eye Scaling)

A step further in the alignment process is to use the eye distance relative to the reference sample for scaling. The eye distances of all samples are known because the eye locations are landmarked in the CUAVE56-L database as well. This approach bases on the fact that the faces in the database actually do still vary in size: although the subjects were recorded all standing approximately at the same distance from the camera, different people present different mouth sizes. Also, with regard to the potential use of other, additional databases for training, implementing and investigating this scaling modification seems important.

The rotation matrix scale component $\Sigma$ therefore is set to the scale-correction factor multiplied with $I_2$, as equation 5.2 shows. In this equation, $d_{eyeRef}$ is the eye distance from a reference sample, $d_{eye}$ is the eye distance of the current sample to be aligned. Since only similarity transformations were used, the scale factor can be multiplied at any time, as shown at the end of the equation.

$$\hat{R}_{EyeScale} = U \left( \frac{d_{eyeRef}}{d_{eye}} I_2 \right) V^* = \frac{d_{eyeRef}}{d_{eye}} \cdot \hat{R}_{NoScale}$$  \hspace{1cm} (5.3)

Alternative Rotation Method using Mouth Corners (Eye Scaling MCR)

Since the translation part of the alignment process must be done manually in either case - the center of mass of each shape is moved to the origin (0,0), only the rotation part of the Procrustes analysis described in the appendix of Cootes report [29] is still used in the EyeScale solution. Since the rotation part of this Procrustes analysis sometimes also produced slanted mouths for incomprehensible reasons, it was assumed that just rotating the shape such that the mouth corners lie on a horizontal line may give a better alignment. Of course this will only work well if the mouths themselves are actually symmetric, but as long as people in the training set are not “making faces”, this can be assumed to be given to the required extent. Equations 5.4 to 5.6 show how the rotation matrix is calculated in this case.

\(^1\)Two-dimensional scaling: the rotated shape can be scaled independently in x and y direction.
Chapter 5. Model Variations & Parameters

\[ \vec{m} = \vec{p}_{\text{leftMouthCorner}} - \vec{p}_{\text{rightMouthCorner}} \]  (5.4)

\[ \theta_m = -\arctan\left(\frac{m_y}{m_x}\right) \]  (5.5)

\[ \hat{R}_{\text{EyeScalingMCR}} = \frac{d_{\text{eyeRef}}}{d_{\text{eye}}} \cdot \begin{pmatrix} \cos \theta_m & -\sin \theta_m \\ \sin \theta_m & \cos \theta_m \end{pmatrix} \]  (5.6)

Alternative Rotation without Rescaling (No Scaling MCR)

To bring more insight about the individual effects of the above variations, this last alignment method combines the aforementioned approach that uses the mouth corner for rotation with the idea of omitting the rescaling component (noscale). Therefore, the rotation matrix becomes simply:

\[ \hat{R}_{\text{NoScalingMCR}} = \begin{pmatrix} \cos \theta_m & -\sin \theta_m \\ \sin \theta_m & \cos \theta_m \end{pmatrix} \]  (5.7)

5.3.2 Using Polar Coordinates

A suggestion that also has been investigated is the use of polar coordinates for the shape model. This is a manifest idea, since the mouth in our case consists of two close-to-elliptical shapes. To build the shape model in polar coordinate space, the aligned sample shape coordinate vectors need to be converted before performing the PCA. During the search algorithm the shape needs to be converted back and forth when doing the findFittingShape step (section 3.3). The Cartesian-to-polar coordinate transformations and vice versa are assumed to be generally known and will not repeated here.

There are a few implementation details that need to be considered, but most of all there appear some non-obvious obstacles that are hard to cope with. They are described in the results section along with the experimental results (subsection 7.1.3).

5.3.3 Shape Interpolation

Detection quality increases with the number of points used [28]. Using this as a base for this new approach, we increase the number of shape points by interpolating the lip shapes wherever there is a line in the shape. This works under the assumption that the line in a shape always follows a straight line in the image as well. Since the landmarks in the CUAVE56-L database are rather dense, this can be considered fulfilled. This process is fairly easy, it can be implemented as a preprocessing step to the algorithms.

The interpolation uses a factor \( n_{\text{intp}} \) that specifies how many points per line segment shall be added. The annotated landmark coordinate file then is simply extended by these points. So the initialization and the shape and profile model then treat this data as if the lip actually had been manually landmarked that way. The interpolation is mathematically simple: the vector of each line segment is calculated by subtracting one from the other end point. Then, the vector is split into \( (n_{\text{intp}} + 1) \) of the same length, i.e., the vector is simply divided by this number. We then add this vector one to \( n_{\text{intp}} \) times to the base point of the line segment, obtaining a new point with each
time. Consequently, the new number of shape points is given as \( n_{\text{pts}} = 20 + 20 \cdot n_{\text{intp}} \) with the original lip shape consisting of 20 points.

Figure 5.2 shows an example interpolation. Note that the connections between the inner and outer lip corners are not interpolated. This was omitted for two reasons: first, there is not necessarily a line between those two points, and second, the two points sometimes are very close to each other, so the interpolated sections would be overcrowded with points - basically those points are less suitable landmarks.

### 5.4 Profile Model Variations

#### 5.4.1 2D Profiles

1D profiles as suggested by Cootes were used for most of this project’s testing. The use of 2D patches to model a landmark point’s local appearance was successfully implemented by Milborrow as a mean to further improve detection quality [28]. It is obvious that patches are more suitable for precisely locating corner and point features - what is given in our lip shape at most of the points. This also implies the use of 2D search windows instead of the whiskers used with 1D profiles. We used different search window sizes for mouth corners and normal lip borders, as illustrated in figure 5.3.

Following Milborrow, we do not rotate the patches to align with the shape model whisker. The 2D profiles therefore always stay upright and aligned with the image grid. Without reorientation we must rely on the face being approximately upright, but do avoid interpolation noise. To ensure the model patches scales are correct, all images will be resized so that the eye distance is the same as a given reference (see subsection 5.4.2). Note that the search windows, i.e., the area in which the candidate points for the best local fit lie, are still tilted along the profile whisker directions as shown in 5.3, only the patch templates are kept upright.

It was stated that detection using 2D profiles requires a lot more computation time [28], but since the detection quality with 1D profiles was not overwhelming, this approach was also considered.
In opposition to this statement, in our implementation with Matlab, the use of 2D patches does not necessarily slow down the detection process since we only use integer coordinates (see results subsection 7.3.3). This may be faster because we do not need interpolation as like when extracting 1D profiles.\footnote{Interpolation with 1D profiles is almost mandatory, since the whisker length does not remain constant with different orientations.}

In addition to the already known 2D patches used by Milborrow (Milborrow Patches), we present two modifications, called Gabor Patches and PCA Patches.

### 5.4.1.1 Milborrow Patches

Milborrow’s 2D patches method is a direct translation of the 1D profile method. But instead of a linear gradient, the 2D gradient image is built by convoluting the whole image with different 3x3 gradient filters. After that, the patch around each point is extracted, vectorized and normalized, as done before using 1D profiles.

The profile model then contains the mean patch for each point and the covariance matrix of the vectorized patch, just like in the 1D case.

Furthermore Milborrow describes a couple of tricks that can be applied after the vectorization step, again to both the training samples and the patches extracted during search. For example, the vectors can be equalized using a sigmoid function, that increases the contrast of pixel values around zero. Also, the use of circular and Gaussian masks to weight center pixels more strongly are described. However, those methods seem to have little effect on the detection quality.

### 5.4.1.2 PCA Patches

Principal Component Analysis has already been introduced in subsection 3.2.1 as a statistical method to model the multi-dimensional shape vectors characteristic variations. The general PCA
method however can be used for various other tasks in image processing such as image compression or classification [35]. The PCA patch profile modeling approach bases on the application of PCA for face recognition using so called eigenfaces [36], it was later discovered that Luettin et al. already used the same principle with one-dimensional ASM profiles [25].

The idea is to build an appearance model for each point’s surroundings using PCA, capturing the characteristic intensity value changes caused by different lighting, mouth positions and speaker identities - in the same manner as for the shape model. To achieve this, the patches extracted around the landmark points are simply vectorized and normalized. Then the same procedure can be carried out as for the shape model: find the mean and calculate the covariance matrix’ eigenvectors with the largest corresponding eigenvalues. Therefore, analog to the shape model, we have a mean patch $\bar{p}_k$, an eigenvector matrix $\Phi_k$, each one forming one eigenpatch, and an eigenvalue vector $\tilde{\lambda}_k$ for each of the $n_{pts}$ shape points (20 in our case). Algorithm 6 illustrates this training procedure.
Figure 5.4 shows the first three modes of variation, the first three row vectors of $\Phi_k$.

**Algorithm 6**: The PCA patches training algorithm

**Input**: $n_{smp}$ sample images with $n_{pts}$ shape points each  
**Output**: $\bar{p}_k$, $\Phi_k$, $\bar{\lambda}_k$ for each of the $n_{pts}$ Shape Point $s_k$

1. Extract a square patch $p_i$ around each shape point in every sample.  
2. Vectorize all patches:  
   \[
   p_i (p_{width} \times p_{width}) \rightarrow p_i (p_{width}^2 \times 1)
   \]
3. Normalize all patches:  
   \[
   p_i = \frac{p_i}{\sum |p_i|}
   \]

   **foreach** shape point $s_k \mid k \in [1..n_{pts}]$ **do**

   4.a Calculate mean patch:  
   \[
   \bar{p}_k = \frac{1}{n_{smp}} \sum_{p_i \text{ of } s_k} p_i
   \]

   4.b Calculate Covariance Matrix:  
   \[
   S_k = \frac{1}{n_{smp} - 1} \sum_{p_i \text{ of } s_k} (p_i - \bar{p}_k) (p_i - \bar{p}_k)^T
   \]

   4.c $\bar{\lambda}_k = \text{EigenValues}(S_k)$

   4.d $\Phi_k = \text{EigenVectors}(S_k)$

**end**

During the search algorithm, we determine the best fit of a point by trying to find a PCA patch that looks as similar as possible. That is, for each shape point $k$, we calculate a distance measure for all the candidate patches $\rho$ to a PCA modeled patch. The patches $\rho$ are also vectorized and normalized in the same way as the training patches, so they are basically vectors. To find a model patch that best fits $\rho$, equation 5.8 is used to find a fitting weight vector $b_k$.

\[
\mathbf{b}_k = \Phi_k^T (\rho - \bar{p}_k) \quad (5.8)
\]

We then use the found vector $\mathbf{b}_k$ to calculate the best fitting model patch $\hat{p}_k$, using the model equation 5.9:

\[
\hat{p}_k = \bar{p}_k + \Phi_k \cdot \mathbf{b}_k \quad (5.9)
\]
Figure 5.5: Example of a two-dimensional Gabor filter.

The distance measure used is the euclidean distance between $\rho$ and the model patch $\hat{p}_k$ in the image space, not the PCA space, as shown in equation 5.10.

$$d = \sqrt{\sum_{i=1}^{n_{\text{width}}^2} (\hat{p}_{k,i} - \rho_i)^2}$$

(5.10)

The shape point $k$ is then moved to the center of the patch where the lowest distance was found.

5.4.1.3 Gabor Patches

Gabor filters have been considered a very useful tool in computer vision and image analysis due to their optimal localization properties in both spatial and frequency domain. The impulse response of Gabor filters has been shown to model the visual cortical cells of mammals well and to exhibit excellent selectivity of spatial locality and orientation [37]. Gabor filter-based features have been applied to face recognition and localization or facial expression classification with great success [38, 39, 40]. We therefore investigate Gabor filter features for lip feature point detection.

A Gabor filter is nothing else than a two-dimensional filter, a filter mask that is slided over the image. The filter mask consists of a directed spatial wave multiplied with a Gaussian envelope, as shown in equation 5.11-5.13.

$$g(x, y; \lambda, \theta, \psi, \sigma) = \frac{1}{2\pi\sigma^2} \exp \left(-\frac{\tilde{x}^2 + \tilde{y}^2}{2\sigma^2}\right) \cdot \cos \left(2\pi \frac{\tilde{x}}{\lambda} + \psi\right)$$

(5.11)

$$\tilde{x} = x \cos \theta + y \sin \theta$$

(5.12)

$$\tilde{y} = y \cos \theta - x \sin \theta$$

(5.13)

In this equation $\lambda$ represents the wavelength of the cosine factor, $\theta$ specifies the wave direction and $\psi$ the phase offset at the mask’s center. $\sigma$ is the Gaussian distribution standard deviation and therefore specifies the width of the Gaussian envelope or more generally the size of the mask. Figure 5.5 shows an example Gabor filter mask.
The convolution with the image can also be done in spacial frequency space very easily, which will not be shown here. Furthermore there are multiple alternative equations formulating Gabor filters, many of them including imaginary parts. However, in this work we follow the results of Huang et al. [39], and take only the magnitude of the Gabor filtered images, which results in the equations mentioned above.

For classification, it is intended to run multiple Gabor filter configurations on a single image, which results in multiple filtered images that are taken into account. In the implementation of Gabor filters used in the experiments, we vary the wavelength $\lambda$ and the orientation $\theta$ and the filter size through $\sigma$. The function used takes a number of desired orientations $n_{\text{orient}}$, a minimum wavelength and a number of scales. In each scale, $\lambda$ and $\sigma$ are scaled up by a certain scale factor and $n_{\text{orient}}$ filters are run over the image, each one with different a orientation\footnote{$\theta_k = \frac{\pi k}{n_{\text{orient}}}$ for each orientation $k$ of $n_{\text{orient}}$.}.

During training and search, each input image is convoluted with $n_{\text{orient}} \times \text{scales}$ filters, which results in the same number of what is referred to as Gabor images in this report.

In training (see figure 5.6), a patch around the feature points is extracted from every Gabor image. Every Gabor image patch is then normalized individually, i.e., the sum of absolute values becomes 1 (equation 3.5). The normalization step is very important, without it, our implementation did not work at all.

Then, the patches are vectorized and the individual vectors from each Gabor filter are concatenated into one large feature vector. The feature vectors are merged into one large sample matrix and PCA is performed, as described in the previous subsection.

Within the search algorithm, the best fitting patch is found in a similar way as the last section’s
PCA patches approach. The only difference is that the patches are not extracted from the original image, but from the multiple Gabor images. Regarding speed, it is more expedient to process the frame to be searched by the Gabor filters at the beginning of the algorithm at multiple pyramid levels, not just before collecting the patches around a point. The patches are again normalized and vectorized in the same way as during training. The patch modeling and distance measurement for each candidate location is the same (equations 5.8 - 5.10), but with higher dimensionality.

5.4.2 Image Scale Normalization

For the profiles to be always about the same size relative to the face, the image may be pre-scaled according to the face size. We use the eye distances from the eye detector to rescale the whole image so that the eye distance becomes equal to a reference distance. In our tests, this was 100 pixels, which is approximately the mean eye distance found in the CUAVE56-L database. This size normalization is done before training as well as before search. The search algorithm’s output lip shape consequently must be scaled back to make the shape fit the original input image.

In the experiment we simply collected the outcome of running the algorithms with and without this rescaling step, the results can be found in the results chapter.

5.4.3 Color vs. Grayscale Profiles

Experiments were conducted to investigate if it is appropriate to use color profiles for lip detection. This is worth consideration, since the lips usually have a different color tone than the surrounding skin. The use of color information with ASMs has been successfully applied to locate facial features [41] and non-rigid object tracking [42].

Using color with 1D profiles and with 2D profiles have been investigated separately. With the 1D profiles, the profiles were normalized either on each channel separately, like it was done with the grayscale profiles (equation 3.5) or by using a 3-to-2 channel normalization (RGB2rg). The latter technique normalizes the channels by dividing each channel value by the sum of the three, as shown in equation 5.14 for the red channel. This normalization technique basically drops the overall brightness value, comparable to the ‘value’ component in HSV color space, so this approach relies on color changes on the lip borders.

\[
r = \frac{R}{R + B + G}
\]  

(5.14)

With 2D patches, only single channel normalization was investigated. This means each color channel of a given patch was normalized individually such that the sum of all values becomes 1 (equation 3.5). After that, the channel patches were vectorized and concatenated to form a single feature vector. Consequently this feature vector’s sum of elements is 3. It is noteworthy that carrying out the normalization after concatenation on the whole three-channel vector did not seem to work at all.

The described normalizations must be conducted in the same way with all samples in both training and search - for the search with 1D profiles it is important to note that the normalization must be applied on each sliding window position candidate, and not on the whole profile length.
5.5 Algorithm Variations

5.5.1 Match Weights

The Match Weights approach is an extension of the findFittingShape step in the search algorithm (section 3.3). The idea is to take the match quality of the shape points into account when finding a PCA model shape that fits, i.e., points for which only a poor matching location was found in the findBetterFit step shall be weighted less. We therefore introduce a match weight for each point, which is 1 for good fits and becomes 0 the larger the distance between the model and the best candidate patch is. In our experiments, a linear decrease was used, as illustrated in figure 5.7 or formulated in equation 5.15.

\[ w_k = f(d_k) = \begin{cases} 
1 & d_k < a \\
1 - \frac{d_k - a}{b - a} & a \leq d_k < b \\
0 & d_k > b 
\end{cases} \]  

(5.15)

We want to fully weight all points whose matches are considered good and ignore the bad ones - since this transition is not sharp, the decreasing slope is used. In the above equation, \( a \) and \( b \) are the boundaries of this slope. Their values were determined empirically: first a few real-life point-to-model distances were collected to determine the plausible value range. Then the localization error was measured with different parameter value configurations, and the best performing was selected.

Note that it is not wise to weight the points of a shape relative to each other, i.e., giving the best fitting point a match weight 1 and the worst one 0, because in the case of the fits being all good, the worse fitting shape points would be neglected needlessly. With the linear decrease function on the other hand, if all the matches are bad, the shape will not move at all, but this case is not to be expected.

The match weights \( w_k \) then must be taken into account during the PCA model shape instance generation, that is supposed to fit the best-fit shape given. This is done by using the weights vector \( w \) as given in equation 5.16, when calculating the PCA space shape representation \( b \).

\[ \mathbf{b}_{\text{weights}} = \Phi^T \cdot \Delta \]  

(5.16)

\[ \Delta_k = \tilde{w}_k \cdot (y_k - \bar{x}_k) \]  

(5.17)

In other words, the difference between each mean shape point \( x_k \) and the corresponding best-fit point \( y_k \) is weighted by \( w_k \). This will move the well matched points to their locations as well as possible, and the badly matched will stay close to their mean shape’s position. The weights vector \( w \) needs to be repeated once, i.e., \( \tilde{w} = [w, w]^T \), since each point’s weight needs to be applied to both \( x \) and \( y \) coordinates in the feature vectors. This is not to be confused with the shape vectors \( y \) and \( \bar{x} \).

Furthermore, if the boolean flags “found” are as convergence measure, points with low match weights should be marked as not found, since they are very likely to be moved in the following

---

4 Equation 5.16 then replaces the one the line: Update \( \mathbf{b} = \Phi^T \cdot (\mathbf{y} - \bar{x}) \); in algorithm 4.

5 In Matlab the whole calculation can be done easily by using the times/.* function: \( \mathbf{b} = \Phi^T \cdot ((\mathbf{y} - \text{mean}) \cdot *([\mathbf{w}; \mathbf{w}])^T) \);
shape fitting step.

5.5.2 Number of Eigenvectors on Different Pyramid Levels

During search, the more eigenvectors are used, the more exactly the PCA model shapes can recreate the shapes found by the `findBetterFit` step. The number of eigenvectors therefore must be reduced, otherwise the PCA model will not be able to correct points that are out of place relative to the rest of the lip shape, like the ones shown in figure 3.4. In simple terms, the more eigenvectors we use, the more misshapen lips may be produced - the model will include more noise.

We try varying the number of eigenvectors used on different pyramid levels, in order to increase detection quality and robustness. The idea is to start off with few eigenvectors on high pyramid levels (low resolution) and increase this number on the lower pyramid levels (high resolution). Using less eigenvectors on high pyramid levels should maintain a shape that is considered a valid mouth shape, which is important because at this stage the points are allowed to move larger distances while the search windows are especially feature-rich - wrong placements are more frequent. On lower pyramid levels, where points are only allowed to be displaced by short distances, correct feature point placement is more likely. There, using more eigenvectors will allow the output shape points to be at their local best-fitting location, which allows more precise placement.

The parameter space that opens when using this approach may easily get vast, since it grows exponentially with each parameter level. For example if we consider $E = \{3..10\}$ as number of eigenvectors and have $n_{pyr} = 3$ pyramid levels, we get $\|E\|^3 = 512$ configurations to test. If we only allow the number on lower levels to be higher, we get 112 test configurations, but the growth still is exponential.\(^6\) This aspect must be well considered when trying to find optimal parameters.

\(^6\)This can be calculated as: 
$$\sum_{i=0}^{\|E\|−n_{pyr}} \prod_{k=1}^{n_{pyr}−1} \|E\| − i − k$$
Chapter 5. Model Variations & Parameters

<table>
<thead>
<tr>
<th>Channel</th>
<th>Margin 1%</th>
<th>Margin 3%</th>
<th>Margin 5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>36 - 255</td>
<td>60 - 255</td>
<td>74 - 252</td>
</tr>
<tr>
<td>green</td>
<td>0 - 210</td>
<td>20 - 190</td>
<td>29 - 179</td>
</tr>
<tr>
<td>blue</td>
<td>0 - 185</td>
<td>9 - 169</td>
<td>16 - 158</td>
</tr>
</tbody>
</table>

Table 5.1: RGB histogram equalization remapping table. The input ranges of the individual channels given in the table is scaled to the full possible output range [0..255] (uint8).

5.5.3 Image Preprocessing

As a simple improvement step the input images should be preprocessed so that the lip contours come out better. It is naturally difficult to find an equalization algorithm that will robustly do this due to varying lighting and lip colors. What may be said is that the lip boundaries often do not exhibit strong edges that are essential when finding illumination-independent features. As a first step, the image intensity value range (brightness/contrast) should always be stretched to cover the whole possible range, since we do not lose any information by doing this and enhance small intensity differences. With the CUAVE56-L data base images, this is not necessary because the lighting is always similar and the video frames seem preprocessed in this regard. However, this should be considered when implementing ASM lip detectors that should be more robust.

The preprocessing method investigated in this report is a simple channel level equalization heuristic. We try to bring out the contrast in the individual color channels by stretching their histograms, using typical values found in the mouth area (figure 5.8). According to those histograms, the pixel values were linearly remapped as stated in table 5.1, i.e., the smaller is scaled onto the output range to increase that channel’s contrast. The values of those remappings are deducted by using the histograms: The highest 1% of all values shall be mapped to 255 and the lowest 1% shall be mapped to 0, or with percentages 3% and 5% respectively.\(^7\) To find those, the histogram was sorted and the value at the respective percentage of total indices was taken.

5.6 General Parameter Variations

Working with the ASM lip shape localization algorithms has shown the following model and search parameters to be of considerable importance for the output detection quality:

- Number of PDM eigenvectors used (shape model)
- Profile width / patch size
- Search window length / width
- Convergence constraint: portion of points required to be marked ‘found’
- Convergence constraint: the maximum displacement distance where the point is still marked ‘found’

\(^7\)The images used feature 8 bit unsigned data type (uint8), implying a value range from 0 to 255.
Figure 5.8: Left: RGB histograms of all mouth ROI regions in the CUAVE56-L data set. Right: Example original and equalized mouth ROI patch.

- Pyramid factor
- Number of pyramid levels
- 2D Profiles: (PCA) number of eigenvectors used
- Interpolation factor: additional points per line segment
- Match Weights: weighting function $f(d_k)$

Ideally, these parameters should be re-optimized with every change that is made to the model, since their optimal configuration may change. However, since the parameter space to be tested is large and we can not make assumptions about the independence of the individual parameters, it is not feasible in reasonable time to test all parameters each time, even in an automated way. This was already denoted in the subsection on varying numbers of PDM eigenvectors on different pyramid, where the number of possible combinations is already high.

In the effort to optimize the parameters, we therefore assumed that the parameters were independent. Consequently, a single parameter was varied while keeping the other fixed, and the value with the lowest average error was chosen. On the other hand, values based on experience were also sometimes considered sufficient, e.g. the maximum iterations per pyramid level, which does not seem to affect the algorithm largely.

The results of the experiments conducted to find the optimal values of those parameters are not listed in the results chapter, since they lack of general applicability and scientific relevance. This is due to that the optimal values are not generally applicable, because they are largely depending on the training and testing data. Furthermore the number of figures is large, what would glut said chapter.

Therefore, the values found for the final 1D and 2D profile configurations are listed in appendix A.1, accompanied with the numerous plots that illustrate the process of finding the best possible parameters.
Chapter 6

Tracking

As a last step towards a complete lip localization solution for audio-visual speech recognition, tracking of the lips shall be considered. By tracking, we mean merely to localize the lip shape in a sequence of video frames of the same speaker. No particular tracking techniques in combination with ASMs are described in the literature. Mostly, it is regarded sufficient to just process every single image. Nevertheless there are some steps that may be helpful for finding a smoother, faster or more robust tracking, which will be presented in this chapter.

The steps’ evaluation was done using the Talking Face Video, a single video with annotated feature landmarks (see subsection 4.1.2). The implementation used for the tests is Z5c\(^1\), which is using 2D patches and already employs a combination of the improvement techniques found to increase the localization accuracy. Initially, experiments were started with the 1D profile implementation stage X6f, but we switched to the 2D implementation for reasons that will be stated in the following result chapter, subsection 7.3.3. The training data was extended by twelve landmarked frames from the Talking Face Video sequence itself, since the person in this video has his mouth closed most of the time, which is a case not well covered by the CUAVE56-L data set where the speakers’ mouths are usually open forming phonemes.

Again, the m\(_{ae}17\) measure was used for error measurement. The Talking Face Video’s annotated lip shape consists of only 18 feature points, our detector finds a 20 point shape, since it was trained on the CUAVE56-L data. However, the 18 points are corresponding, so the excessive two landmarks (inner mouth corners) were simply removed when comparing the the found shape and ground truth. For each test, a sequence of 1000 frames was used.

6.1 Shape Filtering

First of all, to make the shape detector’s output more robust and less noisy, we may apply a filter to each shape point’s location coordinates. This should be a causal filter, since the final goal is real-time tracking.

We favoured a simple adaptive linear filter over more sophisticated solutions like Kalman filters or particle filters, since the desired performance could be easily reached with this more simple and faster solution. This conclusion has been drawn after some investigations on filtering the

\(^{1}\)The configurations of those implementation stages can be found in the experiment result tables, appendix A.1.
eye locations used for the mouth ROI detector, since the eye detector sometimes produces wrong outputs, especially when blinking. Consequently, the final tracker program filters the eyes and the mouth shape using this method. The adaptive linear filter’s detailed description can be found in appendix B. Due to the lack of time, no comparison between this method and other filtering solutions was made.

### 6.2 Using the Output-Shape of the Previous Frame

The previous video frame’s output shape can be taken as initialization shape for the search in the next frame. This should at least make the search algorithm converge faster, since the shape points are already close to their convergence locations [25]. In our implementation this may also be convenient, since the computationally expensive mouth ROI detector does not need to be run at every single frame. On the other hand, the lips may move at a very fast speed when talking, so it can not always be assumed that the lip shape’s points are very close to the ones to be found in the next frame, which may be even worse if the speaker’s face is moving in the frame (translating motion). There are two particular things that require special consideration when using the old shape, that will now be discussed.

#### Controlling the Shape Drift

The tracking shape’s state must be supervised and controlled, else the shape may drift away if the localization fails completely in a single frame, e.g., if for some reason the lip shape grows over the entire face, the ASM won’t be able to move it back to the lips’ correct location. Another example is shown in figure 6.1, where the lips warped to one mouth corner.

A simple step to limit this drifting is checking the relation of the lip shape in size, point configuration and position towards the eye positions or face frame and reinitializing the shape if it fails those tests.

We used a heuristic method by testing the four criteria that indicate the lip is out of shape, which is mainly using the eye locations. The eye locations are used since their locations are robust and filtered, which is already used for image size normalization using the eye distance (section 5.4.2). The face frame detector seems to exhibit larger noise regarding size and position.

The lip shape will be reinitialized using the mouth ROI detector if one of the following constraints are not fulfilled:

1. No point of the lip point may move out of the Viola-Jones face frame. This is just an initial, very loose boundary.

2. The line from one to the other mouth corner must not be tilted more than $\pm 30^\circ$ relative to the line connecting the eyes, which prevents mouth rotation. This can be calculated with the scalar product of the vectors between those feature points.

3. This same line connecting the mouth corners must not be shorter than 25% and not longer than 160% of the eye distance. This should by far cover all the naturally possible deformations of the mouth, which about the same length as the eye distance with neutral expression.
4. The line connecting the mid-point of the upper and lower lip must be oriented approximately perpendicular to the eye-line. We chose the constraint that the angle between these two lines must be in the range of $[45^\circ, ..., 135^\circ]$. This keeps the upper and lower lip symmetric.

5. The gravity center of the mouth shape must not be too far away from the face’s symmetry axis. This prevents the shape from moving to a mouth corners while shrinking. The symmetry axis is calculated by using the mid-point $\mathbf{m}$ between the eyes and vector $\mathbf{v}$, which is a vector perpendicular to the connecting line between the eyes $\mathbf{v}$. The distance of a point $\mathbf{a}$ to the symmetry axis is calculated as $\delta = \mathbf{v}^T(\mathbf{a} - \mathbf{m})/||\mathbf{v}||$. In our implementation, $\delta$ must not be above one fourth of the eye distance.

The choice of these constraints and their values was not systematically investigated. Therefore the constraint values, like the angles, were intentionally chosen lose. The boundaries could be made more restrictive to produce better output results for a given sequence, since shapes drifting off are intercepted earlier. This was not done during our investigations.

Also, other more sophisticated boundaries can be applied, but the shape drift itself should ideally be low if the lips do not change rapidly.

**Lip Translation Compensation**

Another problem that becomes apparent when only using the last frame’s output shape as initialization is the lip ASMs limited capability to handle horizontal translations. This is illustrated in figure 6.2, where the speaker moves to the right but the detected lip shape only manages follow this motion with a delay. The effect is caused by the distribution of search window or whisker...
orientations: almost all profiles are oriented approximately vertically, since the shape points are on mainly horizontal lines of the lip (see figure 3.3 on page 20), which is even amplified when using shape interpolation. Consequently, the points around the mouth corners are the only ones compensating a horizontal shape mismatch. Because there are few such points, they are not considered enough when finding a PDM instance to fit the points. In other words, if only the points close to the mouth corners move horizontally during local fitting, the rest of the mouth shape will outweigh this movement.

There are multiple ways to counter this effect. One is simply to execute the search algorithm multiple times, while every time the points around the mouth corners are able to pull the shape a little towards the correct horizontal placement. As already mentioned, this method is called stacking according to Milborrow [28]. It is a slow way of achieving better placement, since rerunning the algorithm multiple times obviously takes a lot more time and is not always necessary, i.e., if the face is not moving horizontally.

A faster approach therefore is to use the face frame’s or the eye positions relative motion between two video frames as an indicator for the face’s translational movement. This estimate is then added to the output shape of the previous frame. Alternatively, also the face detector’s frame or even the mouth ROI center can be used to estimate the head’s movement.

As already mentioned, the eye localization is exhibits less noise than the Viola-Jones face detector’s output frame. In our implementation and tests, we therefore used the eye’s movement as indicator for the face’s movement. The results are presented in the following chapter, section 7.3.
Chapter 7

Experimental Results

7.1 Results of Improvement Approaches Testing

This chapter first presents the numerical results of the algorithm variations described in chapter 5. Also, the combination of multiple improvement approaches are discussed. The localization quality is tested as described in section 4.2, i.e., all tests of this section were run on the CUAVE56-L database. In the second section, the results obtained with tracking modifications as described in the previous chapter are presented. A summary of all the results will be given in chapter 8.

7.1.1 Initialization Variation Experiments

Table 7.1 below shows the outcome of the different initialization methods described in section 5.2.

<table>
<thead>
<tr>
<th>Code</th>
<th>Initialization method</th>
<th>average error μ</th>
<th>pseudo-standard deviation std</th>
</tr>
</thead>
<tbody>
<tr>
<td>REF2</td>
<td>VJ Mean Position</td>
<td>8.903</td>
<td>3.250</td>
</tr>
<tr>
<td>IM1</td>
<td>VJ-Scaling, mROI Positioning</td>
<td>8.002</td>
<td>4.715</td>
</tr>
<tr>
<td>IM2</td>
<td>mROI Stretching</td>
<td>7.983</td>
<td>4.612</td>
</tr>
<tr>
<td>IM3</td>
<td>ED-Scaling, mROI Positioning</td>
<td>7.273</td>
<td>3.059</td>
</tr>
<tr>
<td>IM4</td>
<td>mROI Mean Position</td>
<td>6.732</td>
<td>3.432</td>
</tr>
</tbody>
</table>

Table 7.1: Localization quality values achieved by different initialization methods.

The mouth region-of-interest clearly outperforms all the other initialization methods. The other results show that the mouth ROI positioning is not enough and that the Viola-Jones frame’s size is a unsuitable indicator for the mouth’s size. The last possibility described in the previous chapter 5, which uses the eyes as coordinate system, is not listed in this table, since it performed significantly worse than the VJ mean position solution and therefore was not evaluated systematically. This is surprising due to the misconception that the vertical distance between the mouth and the eye line, as a ratio of the eye distance, is roughly the same among different people - it is not. In the CUAVE56-L the ratio is at mean 1.04, but it varies by ±20%. One fifth of the eye distance is a considerably large distance in the face.
Chapter 7. Experimental Results

Apparently the eye distance gives an acceptable estimate for the mouth size nevertheless, even if the mouth width shows comparably large variations in size (±15%).

7.1.2 Training Sample Alignment Variation Experiments

<table>
<thead>
<tr>
<th>Code</th>
<th>Sample alignment method</th>
<th>Initialization Method</th>
<th>$\mu$</th>
<th>$std$</th>
</tr>
</thead>
<tbody>
<tr>
<td>REF1</td>
<td>Original Procrustes Analysis</td>
<td>VJ Mean Position</td>
<td>9.073</td>
<td>3.776</td>
</tr>
<tr>
<td>A2</td>
<td>Eye Scaling</td>
<td>VJ Mean Position</td>
<td>9.212</td>
<td>3.720</td>
</tr>
<tr>
<td>A3</td>
<td>Eye Scaling + Mouth Corner Rot.</td>
<td>VJ Mean Position</td>
<td>9.121</td>
<td>3.303</td>
</tr>
<tr>
<td>REF2</td>
<td>No Scaling</td>
<td>VJ Mean Position</td>
<td>8.903</td>
<td>3.250</td>
</tr>
<tr>
<td>A4</td>
<td>No Scaling + Mouth Corner Rot.</td>
<td>VJ Mean Position</td>
<td>8.593</td>
<td>3.332</td>
</tr>
<tr>
<td>B0</td>
<td>Original Procrustes Analysis</td>
<td>mROI Mean Position</td>
<td>5.621</td>
<td>2.893</td>
</tr>
<tr>
<td>B1</td>
<td>No Scaling</td>
<td>mROI Mean Position</td>
<td>5.586</td>
<td>2.859</td>
</tr>
<tr>
<td>B3</td>
<td>Eye Scaling + Mouth Corner Rot.</td>
<td>mROI Mean Position</td>
<td>5.552</td>
<td>2.996</td>
</tr>
<tr>
<td>B4</td>
<td>No Scaling + Mouth Corner Rot.</td>
<td>mROI Mean Position</td>
<td>5.336</td>
<td>2.689</td>
</tr>
<tr>
<td>B2</td>
<td>Eye Scaling</td>
<td>mROI Mean Position</td>
<td>5.375</td>
<td>2.820</td>
</tr>
</tbody>
</table>

Table 7.2: Localization quality improvement comparison with different alignment methods.

Table 7.2 shows the results of the experiments conducted in order to improve the training sample alignment method. The individual approaches were described in subsection 5.3.1. The variation of the alignment method was done on two different implementation stages: the first uses the Viola-Jones mean position for the shape initialization, the second uses the mouth ROI mean position. The methods are sorted by the achieved mean error $\mu$, within the respective implementation stages’ results (A/B).

Unfortunately, the results do not give a clear picture about the general suitability of the approaches. We are unable to state which technique is performing the best or give some kind of hit-list. Where in the case of the Viola-Jones initialization the methods with the 'No Scaling' component perform best, they do not score so well in the other implementation stage. The 'Eye Scaling' method even performs worst in the first environment and best in the second.

Nevertheless it can be said that at least the 'No Scaling' methods consequently improves the original Procrustes analysis. It then can be concluded that the method most suitable must be picked depending on the rest of the implementation - the approach most adequate must be found through testing.

In conclusion, the basic possibility of reducing the localization errors by modifying the sample alignment technique has been shown, but no generally best-performing method could be found.
7.1.3 Polar Coordinates Experiment

<table>
<thead>
<tr>
<th>Code</th>
<th>Shape model coordinate system</th>
<th>average error $\mu$</th>
<th>pseudo-standard deviation std</th>
</tr>
</thead>
<tbody>
<tr>
<td>REF2</td>
<td>Cartesian</td>
<td>8.903</td>
<td>3.250</td>
</tr>
<tr>
<td>POL</td>
<td>Polar (improvable)</td>
<td>10.522</td>
<td>8.585</td>
</tr>
</tbody>
</table>

Table 7.3: Localization quality comparison of Cartesian and polar shape models.

Disappointingly, the approach of using a polar coordinate PDM (shape model) shows worse performance than the reference. The results are shown in table 7.3. The reasons for that may lie in the different linearity definitions in polar coordinate space, but there exist a more pragmatic issues that need consideration and that could only be solved partially:

- One problem is that all the angular values of one point always need to be in the same range, i.e., problems occur when there are gaps between the numbers caused by the wrap-around at $\pm \pi$. This is an issue that is surprisingly difficult to solve in a general way. We just made sure that all the angles of a point are not further apart than $\pi$, by adding or subtracting $2\pi$ in those cases.

- In our data set there exist points close to the origin (0,0) of the coordinate space where the samples are aligned. Those points may sometimes be scattered all around it among different samples. This leads to problems when calculating the mean value and the PCA results. E.g., our point lies at $+\frac{\pi}{2}$ in one sample and at $-\frac{\pi}{2}$ in the other. If the points at the same radii $r$, the mean position is actually (0,0) but just taking the mean of both vector components, the mean returned is $(r,0)$. Basically, this leads to at least a corrupted mean shape, but it also influences the PCA.

The two issues are naturally connected with each other. Since we could not come up with a simple solution for those problems within reasonable time, investigations in this field were not continued in favour of other approaches. In the case of resumption of this approach, possible solutions mentioned by Heap and Hogg [43] are suggested for consideration.

7.1.4 Shape Interpolation Experiments

The improvements obtained using the shape interpolation approach are shown in table 7.4.\textsuperscript{1} The mean error is reduced by about relative 6.3% while the std increases significantly (+27%). In fact, the result of the interpolated approach has less outliers with high localization errors, but the well placed shapes are more widely scattered around the mean. Since the runtime of the search algorithm raises with the number of points ($n_{\text{points}} \times n_{\text{intp}}$), the use of this method is debatable, especially when a speed/quality trade-off must be found. Nevertheless, this method may be used for getting very accurate results.

\textsuperscript{1}Since this approach was investigated as one of the last, the reference (X5) is an intermediate result, with a number of improvements already applied. See appendix C for details.
Chapter 7. Experimental Results

<table>
<thead>
<tr>
<th>Code</th>
<th>Initialization method</th>
<th>average error μ</th>
<th>pseudo-standard deviation ( \text{std} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>X5</td>
<td>Standard</td>
<td>5.070</td>
<td>2.024</td>
</tr>
<tr>
<td>X6</td>
<td>Shape Interpolated ((n_{\text{intp}} = 3))</td>
<td>4.758</td>
<td>2.565</td>
</tr>
</tbody>
</table>

Table 7.4: Localization quality increase when using interpolated shapes.

7.1.5 2D Profile Comparison Experiments

<table>
<thead>
<tr>
<th>Code</th>
<th>Profile type</th>
<th>average error μ</th>
<th>( \Delta )</th>
<th>( \text{std} )</th>
<th>( \Delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>REF0a</td>
<td>STASM - 1D - (VJ)</td>
<td>9.266</td>
<td></td>
<td>7.088</td>
<td></td>
</tr>
<tr>
<td>REF0b</td>
<td>STASM - Milborrow 2D - (VJ)</td>
<td>9.040</td>
<td>−.226</td>
<td>7.018</td>
<td>−.070</td>
</tr>
<tr>
<td>IM3</td>
<td>LLASM - 1D - (mROI mean pos.)</td>
<td>6.732</td>
<td></td>
<td>3.432</td>
<td></td>
</tr>
<tr>
<td>P2D1</td>
<td>LLASM - Milborrow 2D</td>
<td>6.891</td>
<td>+.159</td>
<td>2.856</td>
<td>−.576</td>
</tr>
<tr>
<td>P2D2</td>
<td>LLASM - Gabor Patches</td>
<td>6.637</td>
<td>−.095</td>
<td>3.544</td>
<td>+.112</td>
</tr>
<tr>
<td>P2D3a</td>
<td>LLASM - PCA Patches</td>
<td>6.013</td>
<td>−.719</td>
<td>3.434</td>
<td>+.002</td>
</tr>
</tbody>
</table>

Table 7.5: Localization quality comparison of different 2D patch approaches.

Table 7.5 shows the localization accuracies when using multiple different 2D patch algorithms. The reference values of STASM, using mean Viola Jones frame positions as initializations, are also listed. For the other tests (PD2D1-3), the algorithm version that uses mouth ROI mean position initialization was used as starting point (IM3, see section 5.2), so the results do not compare directly to the STASM values produced with Viola-Jones frame initialization.

Disappointingly, the mean error increases slightly when using Milborrow’s 2D gradient patches. But since the gain achieved by switching to 2D patches with STASM is not too big neither (REF0a & REF0b), these results are coherent. On the other hand, the \( \text{std} \) decreases by 16%. However, since using the patches increases the model size and especially the covariance matrices, the 2D algorithm takes a lot more memory and computation time. Therefore the gradient 2D patches overall are less suitable than the 1D profiles.

The two other patch approaches however manage to reduce the average error. While PCA patches exhibit a clear and significant improvement, the Gabor patches improve the mean-patch approach (Milborrow) only marginally, especially in consideration of the elevated pseudo-standard deviation. Again, as the Gabor patches take much more resources than the 1D profiles, their usage is not recommended in practical applications.

Concluding, the PCA patches approach are the only 2D profile approach investigated that significantly improves localization performance without major drawbacks.
7.1.6 Image Scale Normalization Experiment

<table>
<thead>
<tr>
<th>Code</th>
<th>Scale normalization method</th>
<th>average error $\mu$</th>
<th>pseudo-standard deviation std</th>
</tr>
</thead>
<tbody>
<tr>
<td>X6</td>
<td>none</td>
<td>4.758</td>
<td>2.565</td>
</tr>
<tr>
<td>X6e</td>
<td>Eye-Distance Scaling</td>
<td>4.816</td>
<td>2.547</td>
</tr>
</tbody>
</table>

Table 7.6: Localization quality with and without eye-distance scale normalization.

Surprisingly, prescaling the image before the training and detection algorithms so that the eye distance is always equal does increase the mean error, as shown in table 7.6. This, however, is an acceptable result, since the faces in the CUAVE56-L data set are all of approximately the same size in the images. Moreover it shows that the eye distance does not represent the face size excessively well, or to be precise, the shape points’ neighborhood feature scales. After all, the differences in the results can be regarded as random noise, especially because the testing set is rather small (56 samples in total).

7.1.7 Color-Grayscale Comparison Experiments

1D Profiles

<table>
<thead>
<tr>
<th>Code</th>
<th>1D profile type</th>
<th>average error $\mu$</th>
<th>pseudo-standard deviation std</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>Color (channel norm)</td>
<td>10.788</td>
<td>4.548</td>
</tr>
<tr>
<td>REF2</td>
<td>Color (BGR2bg norm.)</td>
<td>8.903</td>
<td>3.250</td>
</tr>
<tr>
<td>REF3</td>
<td>Grayscale</td>
<td>7.905</td>
<td>3.225</td>
</tr>
</tbody>
</table>

Table 7.7: Localization quality comparison of color and grayscale 1D profile types.

Apparently, the grayscale approach performs a lot better than the color approaches. The individual channel normalization performs worse than all the other references. The explanation for this actuality are open, nevertheless it may be stated that regarding representation of gradients, the grayscale profiles seem to perform a better job. This result is largely independent of parameters like the profile length. Therefore for 1D gradient profiles, the use of grayscale-only profiles is encouraged.

2D Profiles

The comparison of grayscale and color profiles in combination with 2D PCA patches is not as clear. Apparently, both methods give about the same results. This manifests itself when varying some model or algorithm parameters: figure 7.1 shows a plot of the localization quality versus the number of PDM eigenvectors used. The plots intersect multiple times, therefore the better solution is dependent on this parameter. There are other parameter plots both showing the results for the grayscale and color case that support this statement to be found in appendix A.1 on parameter optimization.
Chapter 7. Experimental Results

5.1 Number of PDM eigenvectors used
5.2 Average error in %

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Average error $\mu$</th>
<th>Pseudo-standard deviation $std$</th>
</tr>
</thead>
<tbody>
<tr>
<td>IM3 Reference - No Match Weights</td>
<td>6.732</td>
<td>3.432</td>
<td></td>
</tr>
<tr>
<td>MW1 Match Weights</td>
<td>6.312</td>
<td>3.198</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.8: Localization quality increase when using match weights.

Using match weights in the search algorithm increases the lip localization quality, as table 7.8 shows. Since the additional computational effort needed is small, only an additional linear function evaluation and a vector multiplication are needed additionally, those results are very positive, also because this concept may be applied to all kinds of ASMs where shape lines follow a straight edge in the image.

\[2\] This was not measured. But the evaluation of one test set with grayscale patches took around 1:20, the color ones 2:30, which is an increase of 88%.

Figure 7.1: Comparison of grayscale and color PCA patches. The leading method changes when varying the number of PDM eigenvectors used.

The search time with color patches is naturally larger.\[2\] Therefore it does not make sense to use color patches, or color profiles at all.

It is remarkable that the implementation with the individual channel normalization ranked well with PCA patches, using the channel intensity values directly, and worst with 1D profiles, where the channel profile gradients were used.

7.1.8 Match Weights Experiments
7.1.9 Number of Eigenvectors on Different Pyramid Levels Experiments

<table>
<thead>
<tr>
<th>Code</th>
<th>Number of eigenvectors used</th>
<th>average error $\mu$</th>
<th>pseudo-standard deviation $std$</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2</td>
<td>4 on all levels</td>
<td>6.128</td>
<td>2.676</td>
</tr>
<tr>
<td>X3</td>
<td>L3:1 $\rightarrow$ L2:3 $\rightarrow$ L1:5</td>
<td>5.732</td>
<td>2.719</td>
</tr>
</tbody>
</table>

Table 7.9: Localization quality increase when varying number of eigenvectors on different pyramid levels.

The experimental results of a successful example with use of different numbers of eigenvectors on the individual search pyramid levels (L3-L1) is given on the second line of table 7.9. The mean error decreases by relative 6.5%, which is a noteworthy result. Unfortunately, the remaining available time for this master thesis did not allow a more in-depth analysis of parameter configurations, so the 1-3-5 solution is a result of a quick heuristic trial and error. It is also important to note that in many configuration cases, the error also increases. Nevertheless, the result verifies the possible localization improvement, which is valuable, since this approach is very applicable to any kind of ASM.

7.1.10 Image Preprocessing Experiment

<table>
<thead>
<tr>
<th>Code</th>
<th>Pre-equalization (margin width)</th>
<th>average error $\mu$</th>
<th>pseudo-standard deviation $std$</th>
</tr>
</thead>
<tbody>
<tr>
<td>X6f</td>
<td>No Equalization</td>
<td>4.772</td>
<td>2.571</td>
</tr>
<tr>
<td>EQ1</td>
<td>Small (1%)</td>
<td>4.767</td>
<td>2.574</td>
</tr>
<tr>
<td>EQ2</td>
<td>Medium (3%)</td>
<td>4.759</td>
<td>2.597</td>
</tr>
<tr>
<td>EQ3</td>
<td>Large (5%)</td>
<td>4.841</td>
<td>2.581</td>
</tr>
</tbody>
</table>

Table 7.10: Localization quality comparison with and without image pre-equalization.

The above table 7.10 shows the localization errors achieved by preprocessing the images with a channel-histogram remapping function. The increase in localization quality even for the best configuration EQ2 is small (<0.3% relative), so this approach is not improving our algorithm enough to be considered useful. The output quality is rather invariant to those changes of input data, which on the other hand demonstrates the illumination-independence of the algorithm.
7.2 Combination of Improvement Approaches

Figure 7.2: Progression of error reduction when combining working improvement approaches with 1D profiles.

Finally, the investigated approaches that were able to increase the localization quality were combined in one algorithm. Figure 7.2 shows the progress while applying various improvements, but keeping the 1D approach. All the way to the left is the STASM reference, followed by our first implementation (LLASM color). The overall error at the intermediate result all to the right (implementation X6) was almost cut in half, from 9.3% to 4.8% (-48%). The step exhibiting the largest difference is the one where we switch to the mouth ROI initialization.

The 2D patches approaches were investigated after the 1D profiles. They were implemented on a version with 1D profiles which uses mouth ROI mean position initialization already (IM4). This intermediate step of replacing the profile model returned a significant decrease of the localization error. Consequently, we then tried to apply the same other improvement techniques that have already improved the 1D result. It turns out that they are less effective in combination with 2D patches: the average error in the last implementation, after having applied the same three modifications as with 1D patches\(^3\), only decreased by 0.9% - where in the 1D case almost 2% were gained.

Remarkable is also that with 2D patches, the improvement approaches match weights and shape

---

\(^3\)Match weights, shape interpolation and varying the number of eigenvectors used on different pyramid levels.
interpolation do only improve the localization quality if their parameters are tuned very precisely, for example the shape interpolation factor. If the parameter values are selected inappropriately, the error may even increase. This circumstance can be seen in figures A.7 and A.8 in appendix A.1 on parameter optimization. The varying of eigenvectors on different pyramid levels did not succeed at all, because the optimal solution apparently always uses 5 eigenvectors on all pyramid levels (figure A.6). Nevertheless, the 2D solution also exhibits a satisfying overall performance, the error was decreased by relative 45% in comparison to the initial reference.

The precise average errors, the associated std values and other configuration details can be found in the experiment result tables in appendix C.2 and C.3 (implementation codes X1-X6 and Z2-Z4).
7.3 Tracking Results

The following results were produced with the 2D patch implementation of the search algorithm (Z5d) unless not specifically labeled.

7.3.1 Filtering Experiment

Figure 7.4 shows the mean error per frame of filtered and unfiltered shapes. In those two tests, the previous frame’s output shape was not reused, the shape was reinitialized in every frame. The filtering may be conducted off-line, i.e., the filter may be applied to the output shape sequence after all frames have been processed, since the filtered shape is not reused in the localization algorithm. The shape point position filtering effectively reduces outlier shapes, where the detection fails completely. Conveniently, this also reduces the overall error, in our test from 8.4% to 7.4% (-12%). On the other hand, filtering makes the lip shape slower in adapting to real position changes of the lips, introducing a delay. This happens a couple of times since the speaker in the Talking Face Video rapidly changes his facial expression and repeatedly tilts his head.

To compensate this movement, the filtered shape from the previous frame may be translated by a small potion of the detected eye movement. This however seems to rather limit the filtering effect, since the filtered eye positions themsleves still contain some amount of noise that is then added to the shape. In conclusion, the user must find a trade-off that suits the application scenario: if the speaker is moving a lot, the error caused by the delay will outweigh the benefits of reducing the outliers.

7.3.2 Using the Output-Shape of the Previous Frame

As expected, the completely unbounded reuse of the previously processed frame’s shape as initialization runs into problems, as soon as the localization in one frame fails completely. The localization errors over the sequence are shown in figure 7.5. Approximately at frame number 260, the eye detector fails multiple times, which causes the shape to drift off. In this particular case, it shrinks and moves to the chin area. The ASM search algorithm is then unable to move the shape back to
Chapter 7. Experimental Results

the lips, resulting in a very high error for the sequence. By applying the boundaries described in section 6.2, the shape is reinitialized whenever it drifts off for a reason. This happens for a total of 12 times.

By adding the head translation correction to the shape of the previous frame, the average error is again reduced by approximately 0.2% (absolute). The upper plot in figure 7.6 shows the error curves with and without translation correction. Generally, there seems to be less lag, the shape seems to stick more to the speaker’s mouth. We can’t show this by numbers since the outliers are very dominant in the current results. Additionally, we applied an on-line filter to the translated shape, which again reduces the mean error by 0.2% (absolute). The error curve is shown in the lower plot in figure 7.6. The filtering removes some of the sharp error peaks in the curve. Although numerical changes seem small, the reduction of outliers significantly improves the subjective impression of the tracking quality.

Comparing the detailed error values as given in table 7.11, we see that the best solution that is using the shape from the previous frame as initialization (BTF) is still considerably less accurate than the unfiltered frame-per-frame solution (F). However, the subjective impression of the tracking quality is higher, since in the (BTF) error curve has less extreme peaks. Also, in comparison to the filtered solution (FF), the tracking seems more accurate, since there is less lag when the speaker moves.

7.3.3 Error Tables and Runtime Aspects

As mentioned in section 6.2, using the shape from the previous frame as initialization for the next search bears not only the possible advantage of faster convergence, but also removes the need to precisely localize the mouth ROI in each frame. This might be very relevant for practical applications aiming for real-time tracking. As table 7.12 shows, the mean processing time per frame is significantly smaller when reusing previous frame’s output shape and therefore skip the mouth ROI detector. The bounded solution takes the least average computation time. This is coherent to our expectations: first, since the shape always stays approximately in its right place, unlike in the unbounded case, the algorithm converges fast. Second, no costly filtering is processed in each frame.
Chapter 7. Experimental Results

Figure 7.6: Above: translated vs. untranslated reuse of the output shape of the previous frame (B/BT), below: the same configuration, but with an on-line filtering of the translated shape (B/BTF).

<table>
<thead>
<tr>
<th>Description</th>
<th>reuse shape</th>
<th>$\varnothing$ t/frame</th>
<th>mean error</th>
<th>median error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame-per-frame processing (F)</td>
<td>No</td>
<td>4.595 s</td>
<td>8.380 %</td>
<td>6.461 %</td>
</tr>
<tr>
<td>Frame-per-frame processing, filtered (FF)</td>
<td>No</td>
<td>4.595 s</td>
<td>7.436 %</td>
<td>6.281 %</td>
</tr>
<tr>
<td>Unbounded (U)</td>
<td>Yes</td>
<td>1.655 s</td>
<td>23.979 %</td>
<td>27.869 %</td>
</tr>
<tr>
<td>Bounded (B)</td>
<td>Yes</td>
<td>1.356 s</td>
<td>9.653 %</td>
<td>7.585 %</td>
</tr>
<tr>
<td>Bounded, translated (BT)</td>
<td>Yes</td>
<td>1.598 s</td>
<td>9.435 %</td>
<td>7.145 %</td>
</tr>
<tr>
<td>Bounded, translated, filtered (BTF)</td>
<td>Yes</td>
<td>1.548 s</td>
<td>9.262 %</td>
<td>7.190 %</td>
</tr>
</tbody>
</table>

Table 7.11: Localization error and runtime comparison between different tracking implementations using 2D patches.
Chapter 7. Experimental Results

<table>
<thead>
<tr>
<th>Description</th>
<th>reuse shape</th>
<th>$\varnothing \ t$/frame</th>
<th>mean error</th>
<th>median error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame-per-frame processing (F)</td>
<td>No</td>
<td>$&gt;40 \ s$</td>
<td>7.305 %</td>
<td>5.621 %</td>
</tr>
<tr>
<td>Frame-per-frame processing, filtered (FF)</td>
<td>No</td>
<td>$&gt;40 \ s$</td>
<td>6.230 %</td>
<td>5.301 %</td>
</tr>
<tr>
<td>Bounded, translated, filtered (BTF)</td>
<td>Yes</td>
<td>$\sim 38 \ s$</td>
<td>9.898 %</td>
<td>8.753 %</td>
</tr>
</tbody>
</table>

Table 7.12: Localization error and runtime comparison between different tracking implementations using 1D profiles.

As already mentioned, the initial tests were conducted with the 1D profile implementation X6e. Although the first results showed slightly more accurate localizations than the 2D patches, e.g., 7.2% mean error for the unfiltered frame-per-frame processing, the tests take a lot longer. Table 7.12 shows a few results obtained from the 1D profile tracking. Since processing the 1000 frame long test sequence takes over 10h, we switched to 2D profiles for the testing of the tracking algorithms at a certain point.

There is no cogent reason why 1D profiles take more time per frame, in fact, Milborrow reports exactly the opposite: his 2D patch implementations exhibit considerably longer runtimes per image than the 1D profiles [28]. Analyzing our 1D algorithm showed that the Matlab function `improfile` used to extract the profiles from the image is very slow: about 95% of the total runtime per frame was occupied by this function. It was not attempted to optimize this official Matlab function, which is part of the Image Processing Toolbox. However, it is remains uninvestigated why this function uses up so much time. For implementations in other languages or environments, the 1D profile algorithms may still be faster.
Chapter 8

Conclusion

8.1 Results Summary

In the previous results chapter, the numerical evaluation of several improvement approaches have been given and discussed. The following conclusions are made:

• ASM search initialization using the Hough transform-based mouth region-of-interest detector has shown to outperform all the other investigated alternatives for ASM initialization. The mean shape within this ROI gives a solid, accurate, speaker independent starting point for the algorithm. A localization error reduction up to -24% was measured. Using the information from the eye positions and the face location and size has shown to not give an accurate representation of neither location nor size of the mouth.

• Match weights significantly increase the localization accuracy in the 1D case, but only marginally with 2D patches.

• The same applies for interpolating the shapes by adding points on the shape line segments.

• 2D profiles produce more accurate localization results than 1D profiles in direct comparison. However, the two aforementioned conclusions lead to that the 1D profiles in the end outperform the 2D patches. The most suitable 2D patch approach investigated uses PCA patches, that allow modeling of multiple typical appearances of a local feature. The use of Gabor filtered feature patches was not found to be improving the results.

• With 1D profiles, the use of color profiles has been shown to not be of advantage. When using 2D patches, color and grayscale configurations are equally suitable, where the grayscale variant is favoured since it is faster and uses less memory.

• Revising the alignment used in training may help increasing localization quality. Using different sample alignment methods during training has been shown to increase the model quality and by that the localization accuracy. Depending on the case, a different method returns the best results.

• Color channel pre-equalization does not improve the localization quality considerably.
• Using polar coordinates did not succeed.

The combination of multiple approaches resulted in an overall localization error decrease of -48% compared to the reference. This result was produced with an ASM variant that uses 1D profiles. With 2D profiles, a localization error decrease of -45% was achieved.

As a second part of this project, the improved ASM lip localization programs have been extended to a lip tracker. First, frame-per-frame processing was investigated. Filtering was applied to the lip shape, what effectively reduces outliers and decreases the localization error. On the other hand, filtering also introduces a considerable delay when the lips are moving. The filter therefore must be tuned to suit the needs of the application.

Second, issues that need to be considered when reusing the lip shape found in previous frame as initialization for the next have been discussed. Means to counter undesired effects have been presented, implemented and tested. The results allow the following conclusions:

• Bounding the lip shape’s geometry, orientation, size and position relative to the eyes’ positions is an effective mean to control the shape drift.
• Adding the translational movement of the eyes to the previous frame’s output shape and low-pass filtering the shape lightly increases localization quality and reduces shape-drift.
• In our implementation in Matlab, reusing the shape is preferrable, since it reduces the algorithm’s computation time by 66%. The resulting accuracy is lower, but the subjective impression of the tracker’s performance is better.

8.2 Further Improvements

There are still multiple aspects of the described ASM lip detection algorithms that need consideration, evaluation or optimization. First off, for further research on this matter, a larger database of landmarked, talking lips should be available, in order to increase speaker-independence and robustness. Due to timely constrains of this project, some algorithm parameters were not entirely tested, primarily the Gabor filter parameters and the number of eigenvectors on different pyramid levels. Variation of those parameters may further improve the results obtained with the corresponding approaches. Also, using polar coordinates may be reconsidered, if solutions for problems discussed are found, this method may still excel.

In this master thesis, the goal was to increase localization quality, speed issues were mostly not considered. Only if two solutions exhibit similar localization performance, the faster one was pointed out to be superior. The current implementation in Matlab takes up to 40s per frame in certain configurations. In order to achieve real-time operativeness, the algorithms need to become faster. A first and effective step towards this goal is implementing algorithms in a lower level programming language like C++. Additionally, for 2D patches with PCA model, the Lucas-Kanade algorithm may be applied to reduce the time needed to find the best local fit [44].

In the beginning of this project, considerable effort towards bilinear variants of ASM was spent, however with no presentable solutions. Bilinearity in this context means that the ASM’s PDM consists of two separated models: one part models the lip shape’s variation from person to person, the other models the relative shape variation when a given speaker is talking or expressing emotion.
through the lips. This may also be applied to the profile model. Successful bilinear models for active appearance models have been reported [45, 46].

The tracker program can be further improved regarding robustness and accuracy. Also, occlusion detection is a desirable feature. The lip localization should be turned off in the occlusion case to not disturb later processing like AVSR. This can be done using a classifier that takes the output shape and the point’s match quality into account, since the face and eyes detectors may still return valid positions.

8.3 Conclusion

This master thesis has demonstrated that the ASM can be well applied to perform accurate, speaker-independent lip contour localization on arbitrary images that contain faces. Also, ASM lip tracking has been successfully implemented. A large number of improvement approaches has been investigated and many have been found to increase robustness and localization quality.

We have shown that there exists a lot of potential for small improvements, which altogether manage to significantly increase the localization. Especially regarding robustness and versatility, steps like these are important if ASMs are to excell in the task of lip localization, which can be used for many applications of natural human-computer interaction.
Appendix A

Parameter Optimization

A.1 Parameter Optimization Plots

The important parameters that influence the algorithms localization performance have already been listed in section 5.6. For a good result, it is naturally important to tune all those parameters.

This section will exhibit a few plots that show improvements achieved during the parameter optimization of the 2D PCA patches implementation \textit{P2D2}. In the following section, example parameter configurations for the 1D and 2D case are listed.
Figure A.1: Optimization of the number of eigenvectors used with 2D PCA patches. Apparently, it does not seem to matter excessively, since the resulting error values only vary by approximately 0.2%. The chosen solution was 10 eigenvectors, which gives the returned result for both color and grayscale patches. With 10 eigenvectors, about 95% of the training patches’ variance can be modeled, what can be seen in figure A.2.

Figure A.2: Cumulative sum of PCA patch eigenvalues $\Lambda(n) = \frac{1}{\sum_{i=1}^{n} \lambda_i} \sum_{k=0}^{n} \lambda_k$. This plot illustrates the amount of variance [0..1] that will be covered when using a certain number of eigenvalues. Each line was produced by a different test sub-set of the CUAVE56-L database.
Figure A.3: Optimization of the parameter 2D patch width. Apparently a value around 14 is optimal for modeling the appearance around lips when face size is normalized to have an eye distance of 100 pixels. In this test, 3 pyramid levels were used.

Figure A.4: Optimization of the parameters pyramid level count and pyramid factor. The use of two pyramid levels gives comparable results to the three level case, but in consideration that the search will terminate faster and the profile model is smaller, the two level solution is favoured. Again, the color and grayscale curves intersect multiple times, illustrating their equal performance. The best choice of parameters is two pyramid levels and a pyramid factor of 1.7 (grayscale).
Figure A.5: Cumulative sum of PDM eigenvalues $\Lambda(n) = \sum_{i} \lambda_i \sum_{k=0}^{n} \lambda_k$. This illustrates the distribution of the eigenvalues corresponding to the PDM eigenvectors in matrix $\Phi$. The number of eigenvectors used determines the amount of variation of the training data that can be reproduced by the model. In our implementations we typically used 5 eigenvectors, which represents about 90% of the training data variation.
Figure A.6: Optimization of the number of eigenvectors used on different pyramid levels. Pyramid level 2 is the higher resolution one. Apparently, the choice 5-5 gives the best result.
Figure A.7: Optimization of the parameter interpolation factor. I.e. the amount of points added per line segment. A value of two exhibits the lowest average error, resulting in a model of 60 points.

Figure A.8: Optimization of the match weights function $f(d_k)$. Plot showing multiple configurations of function parameters $a$ and $b$ in:

$$w_k = f(d_k) = \begin{cases} 
1 & d_k < a \\
1 - \frac{d_k - a}{b - a} & a \leq d_k < b \\
0 & d_k > b 
\end{cases}$$
### A.2 1D Parameters

The following parameters were used in the implementation stage X6, that achieved the lowest error (4.7%) with 1D profiles. Number of PDM eigenvectors used (shape model):

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
<th>Value used</th>
</tr>
</thead>
<tbody>
<tr>
<td>neigs</td>
<td>number of PDM eigenvectors used</td>
<td>5 / 3 / 1</td>
</tr>
<tr>
<td>pwidth</td>
<td>1D profile length</td>
<td>15 pixels</td>
</tr>
<tr>
<td>-</td>
<td>search window length</td>
<td>30 pixels</td>
</tr>
<tr>
<td>npyr</td>
<td>number of pyramid levels</td>
<td>3</td>
</tr>
<tr>
<td>pyrfac</td>
<td>pyramid factor</td>
<td>1.4</td>
</tr>
<tr>
<td>-</td>
<td>max. point displacement for being marked found</td>
<td>2 pixels</td>
</tr>
<tr>
<td>-</td>
<td>found points required for convergence</td>
<td>90%</td>
</tr>
</tbody>
</table>

### A.3 2D Parameters

The following parameters were used ..., that achieved the lowest error (4.7%) with 1D profiles. Number of PDM eigenvectors used (shape model):

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
<th>Value used</th>
</tr>
</thead>
<tbody>
<tr>
<td>neigs</td>
<td>number of PDM eigenvectors used</td>
<td>5 / 5</td>
</tr>
<tr>
<td>pneigs</td>
<td>number of 2D PCA patch eigenvectors used</td>
<td>10</td>
</tr>
<tr>
<td>pwidth</td>
<td>2D patch size (side length)</td>
<td>15</td>
</tr>
<tr>
<td>-</td>
<td>search window size</td>
<td>x</td>
</tr>
<tr>
<td>npyr</td>
<td>number of pyramid levels</td>
<td>2</td>
</tr>
<tr>
<td>pyrfac</td>
<td>pyramid factor</td>
<td>1.7</td>
</tr>
<tr>
<td>ftol</td>
<td>max. point displacement for being marked found</td>
<td>5 pixels</td>
</tr>
<tr>
<td>-</td>
<td>found points required for convergence</td>
<td>90%</td>
</tr>
<tr>
<td>cop/cow</td>
<td>Match weights function parameters a/b</td>
<td>0.025 / 0.015</td>
</tr>
<tr>
<td>intfact</td>
<td>interpolation factor</td>
<td>2</td>
</tr>
</tbody>
</table>
Appendix B

Adaptive Linear Filter

Our adaptive linear filter applies to a single point location (x,y). The filter bases on the assumption that the point does not move rapidly and that it moves linearly. The two coordinate components are treated independently, equations B.1 and B.2 show the filter definition for the x component.

\[
\hat{x}_{t+1} = w \cdot x_{t+1} + (1 - w) \cdot \hat{x}_t \\
w = \exp\left(\frac{|x_{t+1} - \hat{x}_t|}{\sigma_t + \alpha}\right)
\]

The first line expresses the linear core of this filter: The new estimate value of x is \(\hat{x}_{t+1}\), which is calculated as a weighted sum of the new sample value \(x_{t+1}\) and the old estimate \(\hat{x}_t\). Note that the weights sum up to 1, and that the weighting factor \(w\) is always between 0 and 1. The larger the difference between the new value and the estimate, the lower the new sample value is weighted. The filter is adaptive to the sample value variance, because \(\sigma_t\) is adapted after each update of \(\hat{x}\), according to equation B.3.

\[
\sigma_{t+1} = \begin{cases} 
\kappa \sigma_t & \sigma_t < |x_{t+1} - \hat{x}_t| + \alpha \\
\frac{1}{\kappa} \sigma_t & \sigma_t > |x_{t+1} - \hat{x}_t| + \alpha \cap \sigma_t > \sigma_{\text{min}} \\
\sigma_t & \sigma_t \leq \sigma_{\text{min}}
\end{cases}
\]

In other words, the variance gets bigger by factor \(\kappa\) each time the difference between the new value and the estimate is larger than the variance itself, and it is reduced by factor \(\frac{1}{\kappa}\) otherwise. This provides a fast, self-controlling mean of the filtering that filters out outlier values reliably. Lastly, the parameter \(\alpha\) is a small variation bonus that is always added to give a static lower limit to the filter’s speed in the case of repeated static sample values. On the other hand \(\sigma_t\) should also be given a lower bound \(\sigma_{\text{min}}\), e.g., 2 pixels. This filter can be easily extended be by adding the first or even second derivative as a further estimator \(\hat{v}\) with the same equation as for \(\hat{x}\) and \(v_t = x_t - x_{t-1}\).
Appendix C

Experiment Result Tables

C.1 Result Table Abbreviations

eS Samples scaled to eye distance
g-sob-mil Grayscale / Sobel-Gradient / Milborrow: mean+cov
IM Initialization variation
meanpos Mean position (within a frame)
mcenter Mouth ROI center
mROI Mouth ROI
myrot Rotate mouth corners to lie on a horizontal line
myrotns Combination of myrot and noscale alignment
myalign Combination of myrot and eS alignment
neigs Number of eigenvectors used
noscale Do not scale the sample point clouds
proc Procrustes analysis
REF Reference test
ROIsc Scale shape over 90% of mouth ROI
ROImp Mean position in mouth ROI
V.Imp Mean position in Viola-Jones face detector output frame
x data missing
1D-g-n 1D profile, grayscale, normalized
1D-c-bg 1D profile, color, 2-channel norm (blue/green)
1D-c-3 1D profile, color, 3-channel norm (rgb)
2D-g-PCA 2D profile, grayscale, normalized, PCA
2D-RGB-PCA 2D profile, color, 3 channel normalized, PCA
### Appendix C. Experiment Result Tables

#### C.2 General 1D Profile Experiments

<table>
<thead>
<tr>
<th>CODE</th>
<th>Description</th>
<th>INIT</th>
<th>SHAPE</th>
<th>PROF</th>
<th>Avg. Error</th>
<th>Std Smp</th>
<th>unnormalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>REF 0a</td>
<td>Milborrow A</td>
<td>VJmp</td>
<td>-</td>
<td>1D-g-n</td>
<td>9.266</td>
<td>7.088</td>
<td>9.1584</td>
</tr>
<tr>
<td>REF 0b</td>
<td>Milborrow B</td>
<td>VJmp</td>
<td>-</td>
<td>2D-g-n</td>
<td>9.039</td>
<td>7.018</td>
<td>9.387</td>
</tr>
<tr>
<td>REF 1</td>
<td>myref A</td>
<td>VJmp</td>
<td>proc</td>
<td>1D-c-bg</td>
<td>9.072</td>
<td>3.775</td>
<td>9.094</td>
</tr>
<tr>
<td>REF 2</td>
<td>myref B</td>
<td>VJmp</td>
<td>noscale</td>
<td>1D-c-bg</td>
<td>8.902</td>
<td>3.249</td>
<td>8.927</td>
</tr>
<tr>
<td>REF 3</td>
<td>grayscale</td>
<td>VJmp</td>
<td>noscale</td>
<td>1D-g-n</td>
<td>7.904</td>
<td>3.225</td>
<td>8.056</td>
</tr>
<tr>
<td>IM2</td>
<td>mROI scale (Ref2)</td>
<td>ROIscale</td>
<td>noscale</td>
<td>1D-c-bg</td>
<td>7.982</td>
<td>4.611</td>
<td>8.039</td>
</tr>
<tr>
<td>IM3</td>
<td>mcenter+eyescaler</td>
<td>mCeS</td>
<td>noscale</td>
<td>1D-c-bg</td>
<td>7.927</td>
<td>3.058</td>
<td>7.308</td>
</tr>
<tr>
<td>IM1</td>
<td>mcenter+VJscale</td>
<td>mCVJ</td>
<td>noscale</td>
<td>1D-c-bg</td>
<td>9.072</td>
<td>3.720</td>
<td>9.207</td>
</tr>
<tr>
<td>F1</td>
<td>scale cloud 2 eyest</td>
<td>VJmp</td>
<td>eS</td>
<td>1D-c-bg</td>
<td>9.111</td>
<td>2.718</td>
<td>5.746</td>
</tr>
<tr>
<td>POL</td>
<td>polar coordinates</td>
<td>ROIscale</td>
<td>polar</td>
<td>1D-c-bg</td>
<td>10.522</td>
<td>8.584</td>
<td>10.620</td>
</tr>
<tr>
<td>F2</td>
<td>vary neigs: 1-3-8</td>
<td>mCVJ</td>
<td>noscale</td>
<td>1D-c-bg</td>
<td>8.962</td>
<td>3.206</td>
<td>8.873</td>
</tr>
<tr>
<td>F3</td>
<td>vary neigs: 012-4-5</td>
<td>mCVJ</td>
<td>proc</td>
<td>1D-c-bg</td>
<td>8.902</td>
<td>6.206</td>
<td>8.950</td>
</tr>
<tr>
<td>F4</td>
<td>vary neigs: 012-4-5</td>
<td>mCVJ</td>
<td>noscale</td>
<td>1D-c-bg</td>
<td>8.900</td>
<td>6.088</td>
<td>8.950</td>
</tr>
<tr>
<td>F5</td>
<td>match-weights</td>
<td>ROImp</td>
<td>noscale</td>
<td>1D-c-bg</td>
<td>6.312</td>
<td>3.198</td>
<td>6.308</td>
</tr>
<tr>
<td>X0</td>
<td>&amp;shapeinterpol: 4x</td>
<td>ROImp</td>
<td>noscale</td>
<td>1D-g-n</td>
<td>5.171</td>
<td>2.631</td>
<td>5.237</td>
</tr>
<tr>
<td>X1</td>
<td>match-weights</td>
<td>ROImp</td>
<td>noscale</td>
<td>1D-c-bg</td>
<td>6.312</td>
<td>3.198</td>
<td>6.308</td>
</tr>
<tr>
<td>X2</td>
<td>match-weights-grayscale</td>
<td>ROImp</td>
<td>noscale</td>
<td>1D-g-n</td>
<td>6.28</td>
<td>2.676</td>
<td>6.172</td>
</tr>
<tr>
<td>X3</td>
<td>match-weights-neigs: 1-3-5</td>
<td>ROImp</td>
<td>noscale</td>
<td>1D-g-n</td>
<td>5.73</td>
<td>2.718</td>
<td>5.764</td>
</tr>
<tr>
<td>X4</td>
<td>match-weights-neigs: 3-3-5</td>
<td>mCVJ</td>
<td>noscale</td>
<td>1D-g-n</td>
<td>5.90</td>
<td>2.595</td>
<td>5.980</td>
</tr>
<tr>
<td>X5</td>
<td>mw-n135-mouthcornerROT</td>
<td>ROImp</td>
<td>myrot</td>
<td>1D-g-n</td>
<td>5.070</td>
<td>2.023</td>
<td>5.396</td>
</tr>
<tr>
<td>X6</td>
<td>&amp;shapeinterpol: 4x</td>
<td>ROImp</td>
<td>myrot</td>
<td>1D-g-n</td>
<td>4.814</td>
<td>2.548</td>
<td>4.8755</td>
</tr>
<tr>
<td>X6b</td>
<td>x:6 + filtermask: [1 0 -1]</td>
<td>ROImp</td>
<td>myrot</td>
<td>1D-g-n</td>
<td>4.757</td>
<td>2.565</td>
<td>4.820</td>
</tr>
<tr>
<td>X6c</td>
<td>back2color, no weights</td>
<td>ROImp</td>
<td>myrot</td>
<td>1D-g-n</td>
<td>6.957</td>
<td>2.9126</td>
<td>7.0756</td>
</tr>
<tr>
<td>X6e</td>
<td>EYESCALE: Whole Image</td>
<td>ROImp</td>
<td>myrot</td>
<td>1D-g-n</td>
<td>4.8165</td>
<td>2.5474</td>
<td>4.8772</td>
</tr>
<tr>
<td>X6f</td>
<td>No Equalization Ref.</td>
<td>ROImp</td>
<td>myrot</td>
<td>1D-g-n</td>
<td>4.771</td>
<td>2.5707</td>
<td>4.8395</td>
</tr>
<tr>
<td>C0</td>
<td>back2color, vjinit</td>
<td>VJinit</td>
<td>proc</td>
<td>1D-c-3</td>
<td>10.789</td>
<td>4.547</td>
<td>10.873</td>
</tr>
<tr>
<td>A2</td>
<td>EyeScale</td>
<td>VJmp</td>
<td>eS</td>
<td>1D-c-bg</td>
<td>9.211</td>
<td>3.720</td>
<td>9.207</td>
</tr>
<tr>
<td>A3</td>
<td>MyAlign</td>
<td>VJmp</td>
<td>myalign</td>
<td>1D-c-bg</td>
<td>9.912</td>
<td>3.302</td>
<td>9.204</td>
</tr>
<tr>
<td>A4</td>
<td>MyRod – NoScale</td>
<td>VJmp</td>
<td>myrot</td>
<td>1D-c-bg</td>
<td>8.592</td>
<td>3.331</td>
<td>8.631</td>
</tr>
<tr>
<td>B0</td>
<td>Procrustes</td>
<td>mROI</td>
<td>proc</td>
<td>1D-g-n</td>
<td>5.621</td>
<td>2.893</td>
<td>5.641</td>
</tr>
<tr>
<td>B1</td>
<td>NoScale</td>
<td>mROI</td>
<td>noscale</td>
<td>1D-g-n</td>
<td>5.586</td>
<td>2.859</td>
<td>5.613</td>
</tr>
<tr>
<td>B2</td>
<td>MyRod – No Scale</td>
<td>mROI</td>
<td>myrot</td>
<td>1D-g-n</td>
<td>5.552</td>
<td>2.995</td>
<td>5.593</td>
</tr>
<tr>
<td>B3</td>
<td>EyeScale</td>
<td>mROI</td>
<td>myrot</td>
<td>1D-g-n</td>
<td>5.375</td>
<td>2.820</td>
<td>5.404</td>
</tr>
<tr>
<td>B4</td>
<td>MyAlign</td>
<td>mROI</td>
<td>myalign</td>
<td>1D-g-n</td>
<td>5.336</td>
<td>2.689</td>
<td>5.363</td>
</tr>
<tr>
<td>EQ1</td>
<td>Slight ~1%</td>
<td>ROImp</td>
<td>myrot</td>
<td>1D-g-n</td>
<td>4.7669</td>
<td>2.5742</td>
<td>4.833</td>
</tr>
<tr>
<td>EQ2</td>
<td>Medium ~3%</td>
<td>ROImp</td>
<td>myrot</td>
<td>1D-g-n</td>
<td>4.7585</td>
<td>2.597</td>
<td>4.8314</td>
</tr>
<tr>
<td>EQ3</td>
<td>Large ~5%</td>
<td>ROImp</td>
<td>myrot</td>
<td>1D-g-n</td>
<td>4.8409</td>
<td>2.5806</td>
<td>4.9144</td>
</tr>
</tbody>
</table>
## C.3 2D Profile Experiments

<table>
<thead>
<tr>
<th>CODE</th>
<th>Description</th>
<th>INIT</th>
<th>SHAPE</th>
<th>PROF</th>
<th>Avg. Error</th>
<th>Std Smp</th>
<th>unnormalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>REF X</td>
<td>Just initialize</td>
<td>ROImp</td>
<td>-</td>
<td>-</td>
<td>7.300</td>
<td>2.731</td>
<td>7.394 2.978</td>
</tr>
<tr>
<td>P2D1</td>
<td>2D Milborrow Redux</td>
<td>ROImp</td>
<td>noscale</td>
<td>g-sob-mil</td>
<td>6.891</td>
<td>2.856</td>
<td>6.905 2.842</td>
</tr>
<tr>
<td>P2D2</td>
<td>2D-Gabor-nPCA</td>
<td>ROImp</td>
<td>noscale</td>
<td>2DG:2s/4o</td>
<td>6.636</td>
<td>3.526</td>
<td>6.733 3.619</td>
</tr>
<tr>
<td>P2D3b</td>
<td>2D-RGB-PCA</td>
<td>ROImp</td>
<td>noscale</td>
<td>2D-RGB-PCA</td>
<td>5.952</td>
<td>3.19</td>
<td>6.002 3.073</td>
</tr>
<tr>
<td>P2D3a</td>
<td>2D-gray-PCA</td>
<td>ROImp</td>
<td>noscale</td>
<td>2D-g-PCA</td>
<td>6.013</td>
<td>3.434</td>
<td>6.045 3.255</td>
</tr>
<tr>
<td>Z2</td>
<td>optimized</td>
<td>ROImp</td>
<td>noscale</td>
<td>2D-g-PCA</td>
<td>5.396</td>
<td>2.705</td>
<td>x     x</td>
</tr>
<tr>
<td>Z3</td>
<td>shape interpolation</td>
<td>ROImp</td>
<td>noscale</td>
<td>2D-g-PCA</td>
<td>5.277</td>
<td>2.764</td>
<td>x     x</td>
</tr>
<tr>
<td>Z4</td>
<td>match weights</td>
<td>ROImp</td>
<td>noscale</td>
<td>2D-g-PCA</td>
<td>5.099</td>
<td>2.751</td>
<td>5.178 2.827</td>
</tr>
<tr>
<td>Z5a</td>
<td>MyAlign</td>
<td>ROImp</td>
<td>myalign</td>
<td>2D-g-PCA</td>
<td>5.232</td>
<td>2.719</td>
<td>5.309 2.784</td>
</tr>
<tr>
<td>Z5b</td>
<td>MyRot</td>
<td>ROImp</td>
<td>myrot</td>
<td>2D-g-PCA</td>
<td>5.175</td>
<td>2.753</td>
<td>5.244 2.803</td>
</tr>
<tr>
<td>Z5c</td>
<td>eyescale</td>
<td>ROImp</td>
<td>eS</td>
<td>2D-g-PCA</td>
<td>5.238</td>
<td>2.708</td>
<td>5.317 2.777</td>
</tr>
</tbody>
</table>
Bibliography


