Facial Feature Extraction Using a Probabilistic Approach

Mustafa Berkay Yılmaz[a], Hakan Erdoğan[a,*], Mustafa Ünel[a]

[a] *Faculty of Engineering and Natural Sciences, Sabancı University, Orhanlı, Tuzla, 34956, Istanbul, Turkey*

Abstract

Facial features such as lip corners, eye corners and nose tip are critical points in a human face. Robust extraction of such facial feature locations is an important problem which is used in a wide range of applications. In this work, we propose a probabilistic framework and several methods which can extract critical points on a face using both location and texture information. The new framework enables one to learn the facial feature locations probabilistically from training data. The principle is to maximize the joint distribution of location and apperance/texture parameters. We first introduce an independence assumption which enables independent search for each feature. Then, we improve upon this model by assuming dependence of location parameters but independence of texture parameters. We model location parameters with a multivariate Gaussian and the texture parameters are modeled with a Gaussian mixture model which are much richer as compared to the standard subspace models like principal component analysis. The location parameters are found by solving a maximum likelihood optimization problem. We show that the optimization problem can be solved using various search strategies. We introduce local gradient-based methods such as gradient ascent and Newton's method initialized from independent model locations both of which require certain non-trivial assumptions to work. We also propose a multi-candidate coordinate ascent search and a coarse-to-fine search strategy which both depend on efficiently searching among multiple

[*]Corresponding author

Email addresses: `berkayyilmaz@sabanciuniv.edu` (Mustafa Berkay Yılmaz), `haerdogan@sabanciuniv.edu` (Hakan Erdoğan), `munel@sabanciuniv.edu` (Mustafa Ünel)

candidate points. Our framework is compared in detail with the conventional statistical approaches of active shape and active appearance models. We perform extensive experiments to show that the new methods outperform the conventional approaches in facial feature extraction accuracy.

Keywords: facial feature extraction, probabilistic method, gradient-based optimization, search methods, Gaussian mixture models, principal component analysis

---

1. Introduction

Flexible objects provide a challenge to computer vision since they can change their form in unexpected ways and appear differently from one image to another. One promising technique for characterizing and recognizing flexible shapes is determining critical landmark points on the object. Such landmark points can be used to model the shape and appearance of these objects. Human face can be considered as a flexible object and critical points on a face can be easily identified. These critical points can be the lip corners, eye corners and nose tip for example.

In this paper, we call these critical points facial features and our goal is to detect the location of those features. Facial feature extraction is an important problem that has applications in many areas such as automatic visual emotion detection, gesture recognition, pupil tracking, driver fatigue detection for safe driving, face detection systems following a bottom-up approach, facial image compression and low-bit video coding.

1.1. Facial feature extraction methods in the literature

Facial feature extraction from a face image has been an intensive research area. The work in this area can be divided into two parts: rule-based and statistical.

Many works in the literature make use of the symmetry in a human face. In [1], a generalized symmetry operator is used. Some other works following this approach are [2, 3, 4]. Geometry-based approaches use some rule based a-priori information. Some example works incorporating strict geometric models are [5, 6, 7, 8, 9]. Low level image features such as corners and edges are intensively used in the literature. Some previous works relying on those features are [8, 10]. Works especially published recently using low level image features are utilizing a method "Smallest Univalue Segment Assimilating

Nucleus" or its abbreviation SUSAN, proposed in [11]. There are many works using this method to extract low level image features of a face image, for instance [12, 13]. A method which relies on the extraction of low level features will not be robust since there are many candidate points extracted. Typically, a symmetry based method or a geometrical relation is used to eliminate candidate points. This clearly does not give the optimal locations of facial features because of variations such as scaling, translation, rotation and illumination conditions. Any kind of variation will make those a priori rules useless.

Statistical approaches include active shape and apperance models. Shape based approaches usually learn a subspace of facial feature point locations, by introducing a point distribution model. The most famous method under this class of algorithms is active shape models (ASM) also known as smart snakes [14, 15]. Recent works making use of ASMs for facial feature extraction are [16, 17, 18]. Unfortunately, there are some disadvantages of shape based models. They only make sparse use of image information using local appearance features. In some later works such as [18]; some possible improvements are offered for ASM. Those improvements are: fitting more landmarks than actually needed; selectively using two instead of one-dimensional landmark templates; adding noise to the training set; loosening up the shape model as the iterations advance; trimming covariance matrices by setting most entries to zero and stacking two ASMs in series. However, whole appearance information is an important property of human face and it is employed in the active appearance model (AAM). AAM uses the whole apperance of the represented region by warping the current image to a standard shape using locally linear mappings and model the variation of the texture in these warped images. The method is expected to generalize to an unseen example as long as the training database is large enough. AAM is first introduced in [19]. It is based on ASM however there are significant differences. A good comparison of ASM and AAM is given in [20]. Some works following the AAM approach are [21, 22]. Performance of ASM and AAM approaches are dependent on starting point accuracies. They may not directly handle cases well outside of the training set. This includes unseen people in test data, occlusions and extremely deformable objects [23].

There have been some earlier studies on facial feature localization for the purpose of more accurate face detection [24, 25, 26, 27]. The goal in these studies are to roughly locate the facial features and use this information to detect a face. Their goal is much different than ours in this paper. We are aiming for very accurate facial feature localization and measure our success as the deviation from hand-labeled feature points which we use as ground truth. We think that AAM and ASM are the best competitors of our approach being the de facto state-of-the-art methods for this specific problem and there is no indication to believe that other approaches are better alternatives to them. In addition there have been some studies to locate specific features such as eye centers [28] which are in a different category since they specifically target only certain facial features.

1.2. Our approach and relation to earlier work

Approaches like ASM and AAM are widely used for the purpose of facial feature extraction. These are very popular methods, however their accuracies are limited. They may perform poorly on an unseen person's face or face images with accessories or other unexpected variations [23]. Both methods make use of principal component analysis (PCA) which finds a linear manifold (a shifted subspace) that models the data. ASM method utilizes a linear manifold for the location parameters while local appearance on a line perpendicular to the contour is usually modelled with a Gaussian. AAM utilizes linear manifolds (subspaces) of location and holistic appearance parameters which are learned from training data. However, by default, learning in AAM and ASM is not probabilistic and every point in the manifold is considered equally likely. It is possible to use a probabilistic interpretation of PCA to formulate a probabilistic version of AAM. However this probabilistic interpretation of the AAM corresponds to a specific kind of a probabilistic model (factor analysis model with a constant noise variance, so called probabilistic PCA [29]) which is not the only possible probabilistic model one can use. ASM and AAM are preferred in the context of facial features historically since PCA uses a small number of parameters and yields fast and efficient search algorithms. Our formulation of the problem is quite different than ASM and AAM in that we do not assume that the feature point locations lie in a linear manifold of the high dimensional space and we use a more general appearance model.

The assumption of linear manifolds are used when there is strong evidence that such a subspace exists or when there is not enough data to learn all the parameters of a more complex model. In facial feature extraction, this might have been true since recently since there were not many available databases of hand-marked facial features. However, this has been changing recently. Recently, crowdsourcing tools such as Amazon Mechanical Turk

[30] and MIT's LabelMe [31] have enabled cheap acquisition of labeled images for training statistical models. This makes it very important to invest on more accurate machine learning approaches for the problem of facial feature extraction and other landmark point extraction tasks. Since we are experiencing an explosion of data availability, it is possible to build more complex models with a large number of parameters which have a potential to be more accurate than the simpler models with a few parameters. This paper is a work in this direction to be able to make use of more data to train more complex models.

The model we used in this paper basically assumes that the location parameters are represented by a full-covariance Gaussian, and local texture/appearance parameters are represented with a Gaussian mixture model (GMM). No assumption of a linear manifold is used. This makes our model much powerful to represent different appearances of people in the world. The model is developed from scratch using a fully probabilistic formulation which did not exist in the literature before. Hence, it is a completely new formulation for the solution of the problem. Our model, with slightly different assumptions on location and texture parameters and distributions, can be shown to subsume variants of the ASM and AAM models. However, our choices of the location and texture models make our model completely different than the ASM and AAM models with the basic difference being the mis-assumption of the linear manifold and Gaussianity. GMM, which is used to model the texture in our model is much more powerful than probabilistic PCA since it can be used to fit a multiple mode distribution to data.

The difference between PCA and GMM is illustrated in Figure 1 on a 2D dataset. The linear subspace which is found by PCA (dashed line) and the equidensity curves of a three mixture GMM (solid curves) are shown. It is clearly seen that these models are much different from each other and that the GMM model is a much better fit as compared to the PCA model. For example, PCA model will consider that the point shown with a circle is more likely than the point shown with a cross, however the GMM model will assign a higher probability to the cross as compared to the circle which makes more sense intuitively. Although this dataset is in two dimensions, the illustration helps to visualize the picture in higher dimensions as well.

Our approach is a new probabilistic method which is able to learn both texture and location information of facial features in a person-independent manner. The algorithm expects a face image as the input which is the output of a good face detection algorithm. It finds the best facial feature locations

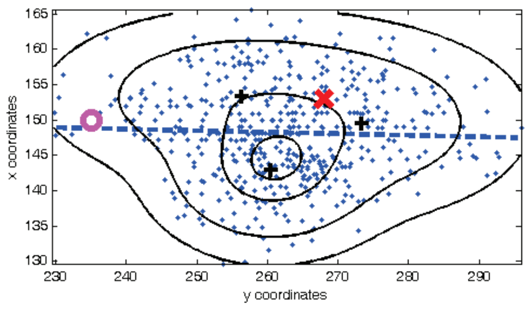


Figure 1: Representation of a set of two dimensional data points by PCA and GMM models. The data are the x and y coordinates of an eye corner from 500 face images. The dashed line shows the one-dimensional PCA subspace, the solid contours represent the three mixture GMM equidensity curves. The models will assign different probabilities to the two points indicated by a cross and a circle.

by maximizing the joint distribution of location and texture parameters and assuming appropriate parametric distributions for them. Initial candidate points are found using an independence assumption easily. Then, we improve upon this model by assuming dependence of location parameters but independence of texture parameters which enables us to model location and texture features separately in a complex model. This new complex problem is solved using some practical optimization or search algorithms since an exhaustive search aiming to find the optimal locations is computationally infeasible. Devising these various search methods is also a novelty of this paper. Although the search methods are well known algorithms, they have not been used in this context before and they require certain modifications and assumptions to work efficiently in this problem. The introduced methods are novel competitive techniques in the problem of finding facial feature point locations. We show that, using our methods, it is possible to find the locations of facial features in a face image with less errors as compared to the ASM and AAM methods which we consider as our main competitors. Extensive experiments are done on three different databases to consistently show the superiority of the proposed methods.

A much shorter version of this work appeared in [32], where we developed the independent and dependent location models for facial feature extraction. In this work, we extend and embellish that previous work by introducing and developing several new search techniques and a detailed discussion of results. We also introduce a new section on texture representation using principal component analysis. Two new search techniques are the coordinate ascent and coarse-to-fine search techniques that will be explained in section 3.

The rest of the paper is organized as follows. Section 2 explains our novel probabilistic model. In section 3, we focus on optimization and search algorithms utilized to solve the optimization problem defined in section 2. Experimental results are presented in section 4. Finally in section 5, we summarize our findings and propose some future improvements.

2. The probabilistic model

Every facial feature is expressed with its location and texture components. Let $\boldsymbol{l}_i = [x_i, y_i]^T$ denote the location of the ith feature in a 2D image[1]. Given a face image and a location $\boldsymbol{l}_i$, we can compute the texture vector $\boldsymbol{t}_i$ associated with it. We use $\boldsymbol{f}_i = [\boldsymbol{l}_i^T, \boldsymbol{t}_i^T]^T$ to denote the overall feature vector of the ith critical point on the face. The dimension of the location vector is 2, and the dimension of the texture vector is p for each facial feature. Define $\boldsymbol{l} = [\boldsymbol{l}_1^T, \boldsymbol{l}_2^T, \ldots, \boldsymbol{l}_N^T]^T$, $\boldsymbol{t} = [\boldsymbol{t}_1^T, \boldsymbol{t}_2^T, \ldots, \boldsymbol{t}_N^T]^T$ and $\boldsymbol{f} = [\boldsymbol{f}_1^T, \boldsymbol{f}_2^T, \ldots, \boldsymbol{f}_N^T]^T$ as concatenated vectors of location, texture and combined parameters respectively.

Our goal is to find the best facial feature locations by maximizing the joint distribution of locations and textures of facial features. We define the joint probability of all features as follows:

$$P(\boldsymbol{f}) = P(\boldsymbol{t}, \boldsymbol{l}). \tag{1}$$

In this study, we will make different assumptions and simplifications to be able to calculate and optimize this objective function. The optimal facial feature locations can be found by solving the following optimization problem:

$$\hat{\boldsymbol{l}} = \text{argmax}_{\boldsymbol{l}} P(\boldsymbol{t}, \boldsymbol{l}). \tag{2}$$

It is not easy to solve this problem without simplifying assumptions. Hence, we introduce some of the possible assumptions in the following section.

[1]The location vector could be three dimensional in a 3D setup

2.1. Independent features model

We can simplify the optimization problem given in equation (2) by assuming independence of each feature from each other. Thus, we obtain

$$P(\boldsymbol{t}, \boldsymbol{l}) \approx \prod_{i=1}^{N} P(\boldsymbol{t}_i, \boldsymbol{l}_i). \tag{3}$$

We can model the joint probability $P(\boldsymbol{t}_i, \boldsymbol{l}_i)$ using a parametric distribution for the concatenated vector $\boldsymbol{f}_i$ and learn the parameters from training data. One choice of a parametric distribution is a Gaussian mixture model (GMM) which provides a multi-modal distribution. With this assumption, we can estimate each feature location independently, so it is suitable for parallel computation. Since

$$\hat{\boldsymbol{l}}_i = \text{argmax}_{\boldsymbol{l}_i} P(\boldsymbol{t}_i, \boldsymbol{l}_i), \tag{4}$$

each feature point can be searched and optimized independently. The search involves extracting texture features for each location candidate (pixels) and evaluating the likelihood function for the concatenated vector $\boldsymbol{f}_i$ at that location. The pixel coordinates which provide the highest likelihood score will be chosen as the seeked feature location $\hat{\boldsymbol{l}}_i$. Although this assumption can yield somewhat reasonable feature points, since the dependence of locations of facial features in a typical face are ignored, the resultant points may be suboptimal in the sense of joint probability.

2.2. Dependent locations model

Another assumption we can make is to assume that the locations of features are dependent while the textures are independent. First, we write the joint probability as follows:

$$P(\boldsymbol{t}, \boldsymbol{l}) = P(\boldsymbol{l})P(\boldsymbol{t}|\boldsymbol{l}). \tag{5}$$

Next, we approximate the second term in the equation above as:

$$P(\boldsymbol{t}|\boldsymbol{l}) \approx \prod_{i=1}^{N} P(\boldsymbol{t}_i|\boldsymbol{l}) \approx \prod_{i=1}^{N} P(\boldsymbol{t}_i|\boldsymbol{l}_i),$$

where we assume that the textures of each facial feature component is only dependent on its own location and is independent of other locations and other textures. Note that, textures of symmetric facial features will be dependent but we ignore this dependence in this work. Since the locations are modeled jointly as $P(\boldsymbol{l})$, we assume dependency among locations of facial features. With this assumption, the equation of joint probability becomes:

$$P(\boldsymbol{t}, \boldsymbol{l}) = P(\boldsymbol{l}) \prod_{i=1}^{N} P(\boldsymbol{t}_i | \boldsymbol{l}_i). \tag{6}$$

We believe this assumption is a reasonable one since the appearance of a person's nose may not give much information about the appearance of the same person's eye or lip unless the same person is in the training data for the system. Since we assume that the training and test data of the system may involve different subjects for more realistic performance assessment, we conjecture that this assumption is a valid one. The dependence of feature locations however, is a more dominant dependence and it is related to facial geometry of human beings. The location of the eyes is a good indicator for the location of the nose tip for example. Hence, we believe it is necessary to model the dependence of locations.

Finding the location $\boldsymbol{l}$ that maximizes equation (2) will find optimal locations of each feature on the face.

2.3. Texture representation using principal component analysis (PCA)

The texture parameters are extracted from rectangular patches around facial feature points. Our probabilistic model does not limit the type of the texture feature used. One may use many different texture features such as Gabor features, local binary pattern (LBP) features or any other gradient-based or gradient histogram based features introduced in the literature. However, we chose to use simple PCA features for simplicity and speed in this work. PCA works quite well for our purposes. To use PCA, we train subspace models from image data inside the rectangular patch and use p subspace basis coefficients as the representation of texture in the patch.

Principal Component Analysis (PCA) is an orthogonal linear transformation that transforms the data to a new coordinate system. PCA can be used for dimensionality reduction by keeping lower-order principal components and ignoring higher-order ones. It is possible to select individual number of lower-order principal components.

We collect the texture data of a facial feature using rectangular patches around that feature's point location, stacking them as column vectors of a data matrix. Each rectangular patch contains M pixels. Suppose $\boldsymbol{X}$ is the mean subtracted data matrix, where each column contains M-dimensional vectors of image pixel intensities. The covariance matrix of mean subtracted data is calculated by

$$\boldsymbol{C} = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^T. \tag{7}$$

An eigenvalue decomposition is applied to the covariance matrix by the formula

$$\boldsymbol{V}^T\boldsymbol{C}\boldsymbol{V} = \boldsymbol{D}, \tag{8}$$

where $\boldsymbol{V}$ is an $M \times M$ square matrix with an eigenvector in each column and $\boldsymbol{D}$ is a diagonal matrix containing the corresponding eigenvalues of eigenvectors. Here, M corresponds to the patch size extracted around a facial feature point. Eigenvectors corresponding to larger eigenvalues are more important for the representation of the data. All of the eigenvectors form an orthonormal basis. Dimensionality reduction is possible ignoring the eigenvectors with lower eigenvalues. Suppose that the dimension is to be reduced to p where $1 \leq p \leq M$, then p eigenvectors with the highest corresponding eigenvalues are placed in columns of the transformation matrix $\boldsymbol{W}$ of size $M \times p$. When we obtain the transformation matrix, it is possible to express the M dimensional feature vector $\boldsymbol{x}_i$ extracted from the patch as p dimensional feature vector $\boldsymbol{t}_i$ by the formula

$$\boldsymbol{t}_i = \boldsymbol{W}^T\boldsymbol{x}_i. \tag{9}$$

The elements of $\boldsymbol{t}_i$ are the expansion coefficients of the orthonormal basis vectors which are directly used as texture features.

2.4. Modeling location and texture features

A multivariate Gaussian distribution is defined as follows:

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)}{(2\pi)^{N/2}|\boldsymbol{\Sigma}|^{1/2}}, \tag{10}$$

where $\boldsymbol{x}$ is the input vector, N is the dimension of $\boldsymbol{x}$, $\boldsymbol{\Sigma}$ is the covariance matrix and $\boldsymbol{\mu}$ is the mean vector.

Probabilistic modeling of location and texture features are separately discussed for independent and dependent models in the following.

2.4.1. Independent features model

For the model defined in 2.1, probability density for each concatenated feature vector $\boldsymbol{f}_i$, $P(\boldsymbol{f}_i)$ is modeled using a mixture of Gaussian distributions. GMM likelihood can be written as follows:

$$P(\boldsymbol{f}_i) = \sum_{k=1}^{K} w_i^k \mathcal{N}(\boldsymbol{f}_i; \boldsymbol{\mu}_i^k, \boldsymbol{\Sigma}_i^k). \tag{11}$$

Here K is the number of mixtures; w_i^k, $\boldsymbol{\mu}_i^k$ and $\boldsymbol{\Sigma}_i^k$ are the weight, mean vector and covariance matrix of the k^{th} mixture component; representing the facial feature point i. $\mathcal{N}$ indicates a Gaussian distribution with specified mean vector and covariance matrix.

2.4.2. Dependent locations model

For the model defined in 2.2, probability density $P(\boldsymbol{t}_i|\boldsymbol{l}_i)$ of texture parameters $\boldsymbol{t}_i$ given location $\boldsymbol{l}_i$ is also modeled using a GMM as in equation (11).

During testing, for each facial feature i, a GMM texture log-likelihood image is calculated as:

$$I_i(x,y) = \log\left(P(\boldsymbol{t}_i|\boldsymbol{l}_i = [x\ y]^T)\right). \tag{12}$$

Note that, to obtain $I_i(x,y)$, we extract texture features $\boldsymbol{t}_i$ around each candidate pixel $\boldsymbol{l}_i = [x\ y]^T$ and find its log-likelihood using the GMM model for facial feature i.

Our model for $P(\boldsymbol{l})$ is a Gaussian model, resulting in a convex objective function. Location vector $\boldsymbol{l}$ of all features is modeled as follows:

$$P(\boldsymbol{l}) = \mathcal{N}(\boldsymbol{l}; \boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{13}$$

Under our assumption of $P(l)$ being multivariate Gaussian, marginal distribution of location parameters of feature i, namely $P(l_i)$, is a two-dimensional Gaussian distribution. Since equidensity curves of a 2D Gaussian is an ellipse, when we threshold this distribution from below, we obtain the inside region of an ellipse in 2D. So, each feature is searched inside an ellipse region which is obtained by thresholding the 2D Gaussian distribution for that feature. These regions are shown in a sample face image in Figure 4. GMM scores are calculated at the centers of each pixel inside these ellipses for faster computation. We assume that we have only N_i candidate points for feature i.

The model parameters are learned from training data using maximum likelihood. Expectation maximization (EM) algorithm is used to learn the parameters for the GMMs [33].

2.5. Relation to ASM and AAM models

As we mentioned in the introduction section, ASM and AAM are also statistical models that can learn from training samples. In this section, we will compare our model with ASM and AAM models. First of all, we must state that ASM and AAM models are not introduced as probabilistic models. Neither are the learning algorithms for ASM and AAM introduced as closed form optimization problems (such as maximum likelihood) of any kind. They are learned by assuming parametric models for shape and appearance, and model parameters are iteratively updated for matching the image at hand. For matching, usually least-squares matching is used which suggests the assumption of a Gaussian noise model (which is not explicitly stated in general).

As mentioned before, the shape model used in ASM and AAM is a PCA model which has a probabilistic equivalent called probabilistic PCA [29]. Probabilistic PCA is a latent variable factor analysis model with constant diagonal noise covariance. This essentially assumes that the data lies in a linear manifold (subspace) which may or may not be a good assumption for the data. This assumption is usually made when there is strong evidence in favor of it. We claim that this model may be a good model for a single person's face, however it is not a good model when a person-independent model is seeked. Thus, we model the shape parameters (l) with a full-covariance Gaussian model.

For the texture part of the model, ASM uses a single dimensional profile model for each feature point $P(t_i|l_i)$ which is modeled as a multivariate Gaussian distribution. Recently, using 2D profiles (similar to our use of 2D patches) has been suggested to improve performance [18]. Our model is a Gaussian mixture model which is an improvement over the single Gaussian assumption.

In AAM, the texture model $P(t|l)$ is equivalent to a probabilistic PCA model in the warped space. In AAM, the texture is modeled as a whole (not individually for each feature) after locally linearly warping the object to a standard shape. This model uses different assumptions about the data than our model. We compare performance of our model with its main competitors ASM and AAM in the experiments section.

Although we show that our model gives better results as compared to ASM and AAM, we would like to point out that AAM and ASM are still useful and important models. For example, AAM model enables generation of the face image with a few number of parameters, so that easy manipulation of faces becomes possible by manipulating the parameters of the model. ASM is also a very fast model which enables fast matching. Our model has more parameters and is more complex to search for the best matching result. However, we foresee that the search speed for our model can be considerably improved by further researches. We also compare the speed of the algorithms in the experiments section. Our goal in this paper is to show that our model is another possible model for facial feature extraction that works considerably well in the databases we worked with. It is a promising approach which moves in a different direction from the mainstream models of AAM and ASM and it is a model that is worth investigating further.

3. Search methods

Brute force search for optimal facial features (which maximize the joint likelihood) will involve an exhaustive search among $\prod_{i=1}^{N} N_i$ candidates where we consider N_i candidates for each facial feature. This is clearly a very large number and we need methods to reduce the amount of computation needed for searching for the best feature locations.

Our first assumption of independence among features enables us to search for each feature separately. For the ith feature, we need only search for N_i candidate locations (pixel centers in the region of interest). We calculate $P(\boldsymbol{f}_i)$ in equation (11) using GMM scores for each candidate location $\boldsymbol{l}_i$ of feature i and decide the location with maximum GMM score as the location for feature i.

For our more advanced dependent-locations model, we develop four different search methods. First two of these methods depend on gradient-based maximization of the dependent-location joint likelihood. The last two search methods are based on efficient search techniques for multiple candidates and use function evaluation only without requirement of calculating (or approximating) the gradient.

We obtain the log-likelihood of equation (6) by taking its logarithm. Because the texture of each feature is dependent on its location, we can define an objective function which only depends on the location vector:

$$\begin{aligned}\phi(\boldsymbol{l}) &= \log\left(P(\boldsymbol{t},\boldsymbol{l})\right) &(14)\\ &= \log\left(P(\boldsymbol{l})\right) + \sum_{i=1}^{N} \log\left(P(\boldsymbol{t}_i|\boldsymbol{l}_i)\right). &(15)\end{aligned}$$

Using the Gaussian model for location and GMM for texture defined in Section 2.4, we can write the objective function ϕ as:

$$\phi(\boldsymbol{l}) = \frac{-\beta}{2}(\boldsymbol{l}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{l}-\boldsymbol{\mu}) + \sum_{i=1}^{N} I_i(x_i,y_i) + C_1. \quad (16)$$

Here, $\boldsymbol{\mu}$ is the mean location vector, and $\boldsymbol{\Sigma}^{-1}$ is the precision (inverse covariance) matrix, learnt during the training. β is an adjustable hyperparameter, allowing us to weight the variance coming from the location model. $I_i(x, y)$ is the score image of feature i defined in equation (12) and C_1 is a constant independent of the location parameters.

So our goal is to find the location vector $\boldsymbol{l}$ giving the maximum value of $\phi(\boldsymbol{l})$:

$$\hat{\boldsymbol{l}} = \text{argmax}_{\boldsymbol{l}}\phi(\boldsymbol{l}). \quad (17)$$

To find this vector, we introduce four different search algorithms and compare their performances in the results section.

3.1. Gradient ascent

We can use gradient ascent to optimize (16) starting from independently found initial locations. Evolution formula for the gradient ascent optimization is given as follows:

$$\boldsymbol{l}^{(n)} = \boldsymbol{l}^{(n-1)} + k_n\boldsymbol{\nabla}\phi(\boldsymbol{l}^{(n-1)}). \quad (18)$$

Here, n denotes the iteration number. We can write the location vector $\boldsymbol{l}$ as:

$$\boldsymbol{l} = [x_1, y_1, x_2, y_2, ..., x_N, y_N]^T. \quad (19)$$

Then we can write the gradient of ϕ as:

$$\boldsymbol{\nabla}\phi(\boldsymbol{l}) = [\partial\phi/\partial x_1, \partial\phi/\partial y_1, ..., \partial\phi/\partial y_N]^T. \quad (20)$$

For a single feature i:

$$\partial\phi/\partial x_i = \frac{\partial}{\partial x_i}\log P(\boldsymbol{l}) + \sum_{i=1}^{N}\frac{\partial}{\partial x_i}\log P(\boldsymbol{t}_i|\boldsymbol{l}_i), \tag{21}$$

and

$$\partial\phi/\partial y_i = \frac{\partial}{\partial y_i}\log P(\boldsymbol{l}) + \sum_{i=1}^{N}\frac{\partial}{\partial y_i}\log\left(P(\boldsymbol{t}_i|\boldsymbol{l}_i)\right). \tag{22}$$

The gradient for the location part can be calculated in closed form due to the modeled Gaussian distribution and the gradient for the texture part can be *approximated* from the score image using discrete gradients of the score image. Plugging in the values for the gradients, we obtain the following gradient ascent update equation for the algorithm:

$$\boldsymbol{l}^{(n)} = \boldsymbol{l}^{(n-1)} + k_n(-\beta\Sigma^{-1}(\boldsymbol{l}^{(n-1)} - \boldsymbol{\mu}) + \mathbf{G}), \tag{23}$$

where

$$\boldsymbol{G} = \begin{bmatrix} G_x^1(\boldsymbol{l}_1^{n-1}) \\ G_y^1(\boldsymbol{l}_1^{(n-1)}) \\ \cdots \\ G_x^N(\boldsymbol{l}_N^{(n-1)}) \\ G_y^N(\boldsymbol{l}_N^{(n-1)}) \end{bmatrix}. \tag{24}$$

Here, G_x^i and G_y^i are the two-dimensional numerical gradients of $I_i(x, y)$ in x and y directions respectively. The gradients are computed only for every pixel center (integers) in the image. Since $\boldsymbol{l}^{(n)}$ is a real-valued vector, we use bilinear interpolation to evaluate gradients for non-integer pixel locations. $\boldsymbol{G}$ is the collection vector of gradients of all current feature locations in the face image. k_n is the step size which can be tuned in every iteration n. Iterations continue until the location difference between two consecutive iterations is below a stopping criterion.

3.2. Newton's method

Similar to the gradient ascent, we may also employ Newton's method which requires calculating the second derivatives with respect to the locations. Iterations for the Newton's method for optimization is given as follows:

$$\boldsymbol{l}^{(n)} = \boldsymbol{l}^{(n-1)} + \boldsymbol{H}_\phi(\boldsymbol{l}^{(n-1)})^{-1}\boldsymbol{\nabla}\phi(\boldsymbol{l}^{(n-1)}), \tag{25}$$

where $\boldsymbol{H}_\phi(\boldsymbol{l}^{(n-1)})$ is the $2N \times 2N$ Hessian matrix of vector function ϕ.

Hessian matrix of the objective function is defined as:

$$\boldsymbol{H}_\phi(\boldsymbol{l}^{(n-1)}) = -\beta\Sigma^{-1} + \boldsymbol{A}, \tag{26}$$

where $\boldsymbol{A} = \frac{\partial^2}{\partial x_i \partial y_i} \sum_{i=1}^{N} I_i(x_i, y_i)$.

It turns out that $\boldsymbol{A}$ is a block diagonal matrix and it can be approximately computed using second order gradients of the score image I_i as follows:

$$\boldsymbol{A} = \begin{bmatrix} G^1_{xx}(\boldsymbol{l}^{n-1}_1) & G^1_{xy}(\boldsymbol{l}^{n-1}_1) & 0 & 0 & ... \\ G^1_{yx}(\boldsymbol{l}^{n-1}_1) & G^1_{yy}(\boldsymbol{l}^{n-1}_1) & 0 & 0 & ... \\ 0 & ... & & & \\ ... & & & & \\ 0 & 0 & ... & G^N_{xx}(\boldsymbol{l}^{n-1}_N) & G^N_{xy}(\boldsymbol{l}^{n-1}_N) \\ 0 & 0 & ... & G^N_{yx}(\boldsymbol{l}^{n-1}_N) & G^N_{yy}(\boldsymbol{l}^{n-1}_N) \end{bmatrix}. \tag{27}$$

The gradients G^i_{yx} and G^i_{xy} are expected to be the same. However, because of the interpolation process, they may end up having different values. So we take the average $\hat{G}^i_{yx} = \hat{G}^i_{xy} = \frac{G^i_{yx} + G^i_{xy}}{2}$ to overcome this problem.

Limited memory Broyden Fletcher Goldfarb Shanno (L-BFGS)[34] and Levenberg-Marquardt algorithms [35, 36] are two different optimization algorithms that depend on the idea of approximately computing the Hessian matrix or its inverse. We have not considered these algorithms in this work as speed was not a major concern and the Hessian matrices we calculate are only 18×18 making it easy to work with them, but it may be possible to speed-up the algorithm (especially when more facial features are required to be found) by considering such approximations.

3.3. Multiple-peak coordinate-wise search

It is possible to find multiple candidates of optimal feature locations using multiple peaks from the independent model. Gradient-based optimization algorithms given in 3.1 and 3.2 may get stuck in a local maximum as they start from the best independent locations and make a gradient based local search. However, if we consider some number of peaks for each facial feature and find the best combination of them; we may end up in a locations vector giving a better objective function value. Even using a reduced number of peaks for each feature, it may not be possible to exhaustively search for each combination. For example, if we consider only 5 candidates for each of 9 features, this would require $5^9 = 1953125$ possible combinations and function evaluations. In order to avoid the highly expensive exhaustive search, we employ the heuristic search method in Algorithm 1 which searches for the best candidate using a coordinate ascent idea.

Algorithm 1 Coordinate ascent search.

```
1: Pick the locations $\boldsymbol{l}_{i,j}$ of $n_i$ best independent GMM score peaks for each
   facial feature $i$ and candidate $j$, where $1 \leq j \leq n_i$ and $I_i(\boldsymbol{l}_{i,1}) \geq I_i(\boldsymbol{l}_{i,2}) \geq$
   $\ldots \geq I_i(\boldsymbol{l}_{i,n_i})$
2: Initialize the new location collection vector $\hat{\boldsymbol{l}}_i \Leftarrow \boldsymbol{l}_{i,1}$ containing the best
   GMM score peak locations, for each facial feature $i$
3: repeat
4:   $\hat{\boldsymbol{l}}_{old} \Leftarrow \hat{\boldsymbol{l}}$
5:   for $k = 1$ to $N$ do
6:     $i \Leftarrow \sigma(k)$
7:     Update $\hat{\boldsymbol{l}}_i \Leftarrow \boldsymbol{l}_{i,\hat{j}}$ where $\hat{j} = \text{argmax}_j \phi([\hat{\boldsymbol{l}}_1^T, \ldots, \boldsymbol{l}_{i,j}^T, \ldots, \hat{\boldsymbol{l}}_N^T]^T)$
8:   end for
9: until $\hat{\boldsymbol{l}} = \hat{\boldsymbol{l}}_{old}$
```

Note that this algorithm guarantees monotonic increase of the log-likelihood but still does not guarantee global convergence. In this algorithm, $\sigma(.)$ denotes a permutation of the sequence $\{1, \ldots, N\}$ thus it determines the order of processing of each feature. In practice, we may change the order of updating each feature to first process the ones that are more reliably found as compared to others as we discuss more in the results section.

3.4. Coarse-to-fine search

It is a common approach in image processing problems to represent an image in different scales and perform a coarse-to-fine search. There have been studies that exploit this in the context of facial feature extraction [37]. So, we applied this approach to our method as well. We learn different texture models at different scales from our training data. Our scales increase twice in size, so if we have $N_x \times N_y$ images, we start with $N_x/8 \times N_y/8$ images and move up to $N_x \times N_y$ by doubling the size at each step and in total, we have four different scales.

To achieve coarse-to-fine search, the following steps are taken:

1. Apply gradient search for the dependent-locations model at the coarsest level.
2. For each finer level, apply independent search in an $M \times M$ window centered around a pixel that corresponds to the location found in the coarser level.

It is not necessary to benefit from dependent-locations model except for the coarsest level since the number of candidate locations is very small, hence the search is made only around previously found locations for each facial feature. This method reduces the candidates to search for each feature by making use of the estimated location from the coarser level. Only M^2 locations are separately searched for each feature at each scale. Typically we take $M = 3$.

4. Experimental results

In this section, we describe our experiments in detail. We give information about pre-processing performed and databases used. We have performed extensive experiments on three different database setups and compared our methods with recent implementations of AAM and ASM methods which are being used in the community. We also compare the computation times for each method.

4.1. Preprocessing

4.1.1. Face detection

For the detection of the face of interest in an image, the approach in [38] is followed without further improvements. This algorithm is currently state of the art and is based on efficiently extracting Haar-like features and using those features in an Adaboost classification - feature selection framework.

An example video frame from our database and its corresponding detected face image are shown in Figure 2.

4.1.2. Face normalization

Face image has to be processed after face detection to normalize various effects such as scaling, rotation, translation and side illumination. We assume that the face detection output is correct, so there is no translation in face image. Face image is resized to a fixed dimension during both training and test steps to overcome the effects of different scalings.

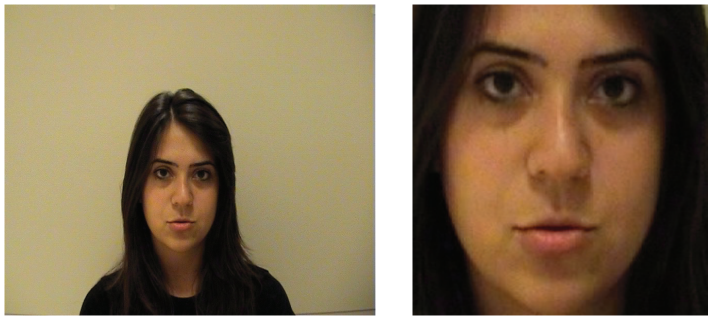

(a) Input image (b) Detected face image

Figure 2: Example face detection result.

To normalize side illumination effects, a four-region adaptive histogram equalization is applied as described in [39]. It is based on dividing the face image into 4 big rectangles and doing histogram equalization in each rectangle separately. Then each pixel is affected by 4 histogram equalization functions by its distance to each 4 regions.

4.2. Databases and evaluation strategy

We used various face video databases and their combinations for comprehensive training and testing of our method. We use three different databases with different train and test images to evaluate our method with four different search techniques and compare the results with the results obtained with ASM and AAM methods which are the main competitors for our method as we elaborated before. We use the standard evaluation criteria used in earlier work which is the mean and maximum obtained over all the test images of m_{e9} distances which is defined as the "average error over the critical points normalized by the distance between the eyes" [18]. In addition to providing the mean and maximum distances, we also provide cumulative error distributions for m_{e9} values for each method as well. In addition we provide search speeds for each method in section 4.8.

Databases that we used for facial feature extraction results are as follows.

1. Sabanci University Turkish Audio Visual Database (SUTAV): Consists of many male and female individuals' frontal videos counting from zero to nine in Turkish. There are two tapes with same individuals and same number of videos. Individuals are in different conditions in two different tapes in terms of dressings, facial hair and illumination. No

excessive head rotations, scalings and translations of the face region is present. It is a relatively challenging database because it includes different illumination conditions. We hand-marked the locations of necessary facial features in selected frames manually to make use of this database.

2. Multi Modal Verification for Teleservices and Security Applications Face Database (M2VTS - [40]): It is made up from 37 different faces and provides 5 shots for each person. These shots were taken at one week intervals or when drastic face changes occurred in the meantime. During each shot, people have been asked to count from zero to nine in their native language (most of the people are French speaking). There are videos including various head rotations. We used the videos without head rotations. We selected random frames per video and hand-marked them as in SUTAV database.
3. Technical University of Denmark Department of Informatics and Mathematical Modeling Face Database (IMMDB - [41]): The IMM Face Database comprises 240 still images of 40 different human faces, all without glasses. The gender distribution is 7 females and 33 males. The following facial structures were manually annotated using 58 landmarks: eyebrows, eyes, nose, mouth and jaw. We used only nine facial features of our interest. Each person has six different images:
   (a) Full frontal face, neutral expression, diffuse light.
   (b) Full frontal face, happy expression, diffuse light.
   (c) Face rotated approximately 30 degrees to the person's right, neutral expression, diffuse light.
   (d) Face rotated approximately 30 degrees to the person's left, neutral expression, diffuse light.
   (e) Full frontal face, neutral expression, spot light added at the person's left side.
   (f) Full frontal face, joker image (arbitrary expression), diffuse light.

   Some facial features are unseen in some face images with much rotation. We found and removed those images.

We used some subsets and combinations of the data- bases explained above and made different experiments with each set:

- Set 1:

  - Training: All IMMDB face images except 24 images having some facial features occluded because of head rotation (216). In addition, randomly selected 4 images of 15 females and 10 males from SUTAV (100). Totally 316 frontal face images are used.
  - Testing: We randomly selected 4 images of 14 females and 11 males from SUTAV (100). No identity in the test set is used in the training set.
  - Purpose: This data set is our base setup where we check the performance of a person-independent system.

- Set 2:
  - Training: In addition to the training part of Set 1, randomly selected images of 27 subjects from each 5 tape of M2VTS are used. 28 face images are eliminated because of faulty face detection results. That makes a training set of 423 face images.
  - Testing: Same as the test part of Set 1.
  - Purpose: This set in comparison to Set 1, can be used to see how increasing the amount of training data increases performance of the algorithms.

- Set 3: Only M2VTS frames are used for this set. Training and testing groups are shown below:
  - Training: 10 random frames from each 5 tape of 10 subjects are selected. Those subjects are distinct from the 27 subjects used in Set 2 training. There are in total 500 face images in this training set.
  - Testing: 2 random frames from each 5 tape of the same 10 subjects are selected. Those frames are distinct from the ones selected in training. There are in total 100 face images in this testing set.
  - Purpose: This set, unlike first two sets, contain face images from the same people in both training and testing data. It is a good set to determine how good the performance is when there is subject overlap in training and testing databases.

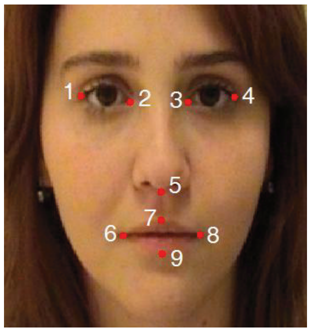


Figure 3: Nine different facial features considered in this work.

4.3. Parameters

In this work, we tried to estimate the locations of nine different facial features. These critical points are four eye corners, nose tip and four lip corners. The numbering of the features are shown on a sample face image in Figure 3. It is possible to use the algorithms developed in this paper to estimate more locations as well, however we found that these nine points are much less ambiguous resulting in more reliable ground-truth data and finding these nine points were enough for our purposes, so it is a waste to search for more points.

There are miscellaneous critical parameters used during our experiments. We found the optimal parameter values with limited experimentation on a subset of Set 1. We used 320×300 face images, then down-sampled them to 80×75 preserving the aspect ratio, for faster computation while improving the performance in terms of errors. We have observed that using higher resolution images resulted in worse results due to algorithms getting stuck at local maxima, so we used a four-fold downsampled image for our search methods except for the coarse-to-fine method which starts with 40×38 and moves up to the full resolution of 320×300. We analyzed the effect of different resolutions in Section 4.6. PCA subspaces of varying dimensions are obtained for different facial features by using the texture information inside rectangular patches around facial features. Four-region adaptive histogram equalization based side-illumination normalization method described in Section 4.1.2 resulted in better facial feature extraction results. We used this method in two different ways: Features having this histogram equalization method as 1; histogram equalization is applied for red, green and blue channels separately and the resulting image is converted to gray-level. For features having

Table 1: PCA subspace dimension and window size parameters used for facial features.

Feature	PCA dimensions	Window size	Hist. eq.
1	30	17×17	2
2	30	17×17	1
3	30	21×21	2
4	30	11×11	2
5	20	11×11	2
6	50	25×25	1
7	50	27×27	2
8	50	25×25	2
9	50	39×39	2

this histogram equalization method as 2; image is converted to gray-level and then histogram equalization is applied to the resulting image. Those training parameters are found experimentally. Optimal PCA dimensions of each facial feature, window sizes used around facial feature points and side illumination normalization methods used are shown in Table 1. For facial features having large variability between different people, like jaw and lip; we had to train larger dimensional PCA subspaces and had to use larger windows.

Search locations for facial features are found by thresholding the marginal Gaussian distributions (derived from $P(l)$) for locations of each feature. Search areas are shown in Figure 4. Ellipses denote the boundaries of search regions and pentagrams denote the mean facial feature locations.

For the independent model explained in Section 2.1, texture coefficients and location vectors are used to build a GMM model to obtain scores. We tried two different types of GMM models. First one is the concatenated vector choice where texture coefficients and location vectors are combined as required in our independent model (GMM_{indep}). In the second choice, we only used texture coefficients as our dependent locations model implies (GMM_{dep}). We used two mixtures for GMMs which gave the best results in most of the experiments which means that there are basically two clusters of texture data for each feature.

For each feature, the pixel giving the highest independent GMM score is

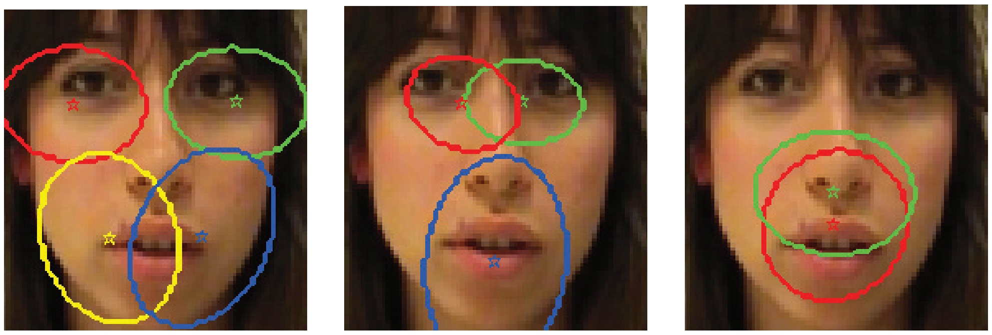

(a) Outer eye and lip corners (b) Inner eye corners and bottom lip corner (c) Nose tip and upper lip corner

Figure 4: Search regions for facial features.

selected as the initial location. These locations are then used to initialize various search methods to solve the dependent locations model introduced in section 2.2. For some cases, locations of the facial features may not converge to optimal locations, and the locations oscillate without further improvement. This situation can sometimes be determined by limiting the maximum number of iterations. If norm of the location difference vector between two consecutive iterations is not lower than a threshold when the maximum number of iterations is reached, optimization process for that image is canceled and independent model result is used as the dependent model result.

In coordinate ascent method, locations are updated with a special order in each iteration. The order is chosen according to the decreasing reliability of GMM scores of facial features. This a priori information is available via the results of previous experiments. Most reliable facial feature is the nose, followed by upper and left-right lip corners, inner eye corners, outer eye corners and bottom lip corner. We have observed that when the system sticks to this order, results are better than any other ordering.

The parameter values such as window sizes, PCA dimensions, GMM mixtures, illumination normalization methods can be fine-tuned for each database using validation data. However, in this paper, we used the indicated values for each database and still obtained good results. This shows that the proposed methods are promising even when non-optimized parameters are used. We also report some results with fixed parameter values in Section 4.7 which are close to the results we obtained in this section.

Table 2: Comparison of median and maximum m_{e9} errors for different search methods considered in this work and other competing methods.

Set 1	IND	GA	NM	CA	C2F	ASM	ASMG	AAM
Med	0.0494	0.0419	0.0416	0.0460	0.0426	0.0566	0.0703	0.0493
Max	0.1449	0.1623	0.1623	0.1462	0.1460	0.1438	0.3681	0.1028
Set 2	IND	GA	NM	CA	C2F	ASM	ASMG	AAM
Med	0.0464	0.0408	0.0403	0.0432	0.0435	0.0539	0.0703	0.0494
Max	0.1424	0.1432	0.1432	0.1600	0.1466	0.1907	0.3681	0.1092
Set 3	IND	GA	NM	CA	C2F	ASM	ASMG	AAM
Med	0.0348	0.0309	0.0308	0.0337	0.0311	0.0405	0.0663	0.0358
Max	0.0772	0.0715	0.0847	0.0902	0.0620	0.1403	0.1017	0.0990

4.4. Accuracy results and comparison

To obtain a performance criteria for testing, we used the method proposed in [37]. Mean of Euclidean point-to-point errors for feature locations is divided by the ground truth inter-ocular distance between the left and right eye pupils. This error is called the normalized error and is applied for the 9 features that we have used, and we name it m_{e9}. We obtain a single m_{e9} value for each face image, which is the average of all facial features' errors. We considered median and maximum of m_{e9} values over all test data as in [18]. The maximum values give an idea about the performance of a method in the worst case scenario. We discuss the performance of our various search methods in different data sets in the following. We also compare the performance of our methods with the ASM method using the ASM implementation STASM [18] and the AAM method using the AAM implementation AAM-API [42].

Results of the independent model (IND), dependent-locations model with gradient ascent (GA), Newton's method (NM), coordinate ascent search (CA) along with coarse-to-fine search (C2F), ASM and AAM are collected in Table 2 for comparison[2]. STASM comes with a pre-built model which is trained using a large data-set. Results obtained with this model is also shown in the same table, denoted as ASMG. Results for the first and second sets are same with the pre-built model since first two sets consist of the same test images.

[2]Some results using the AAM method are slightly different from the results given in [32] because we corrected a computational error in analyzing the results.

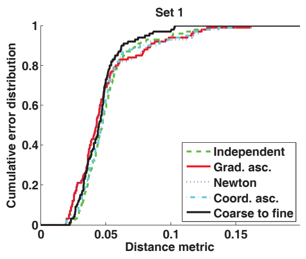
Table 3: Coarse-to-fine search results.

Set 1	40×38	80×75	160×150	320×300
Median	0.0434	0.0454	0.0419	0.0426
Maximum	0.1506	0.1450	0.1480	0.1460
Set 2	40×38	80×75	160×150	320×300
Median	0.0419	0.0448	0.0431	0.0435
Maximum	0.1431	0.1427	0.1464	0.1466
Set 3	40×38	80×75	160×150	320×300
Median	0.0354	0.0346	0.0333	0.0311
Maximum	0.1115	0.0571	0.0595	0.0620

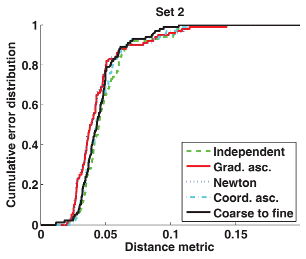
All other methods are trained using the training data for each set. Evolution results of coarse-to-fine search are shown in Table 3. We used 4 different scales to be used in coarse-to-fine search and errors at each resolution are provided. Each finer level searches 3 by 3 pixels around the result of the coarser resolution. We did not use dependent-locations model except for the coarsest level as the candidate pixels at finer resolutions are constrained to be around good locations (achieved by the coarsest level) already and it is a waste to re-evaluate the joint likelihood with various combinations of candidate pixels at each level. So according to our formulation, only coarsest level uses GMM_{dep}. Finer levels use GMM_{indep}.

We also plot cumulative error curves as explained in [37]. On these curves, the value of the y-axis is the ratio of face images with m_{e9} error less than the value in the x-axis. In Figure 5, comparison of cumulative error curve between our methods for all sets are given. In Figure 6, we select our best method for each set and plot its cumulative error curve together with the error curves of ASM and AAM methods.

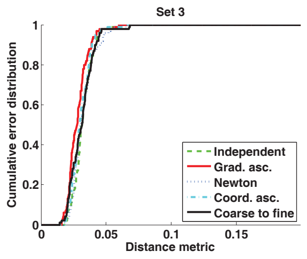
For data sets involving different people in train and test parts, all search methods outperformed ASM and AAM in terms of median errors. For the Set 3 involving same people in train and test parts, our method is still better than AAM, even for the independent model in terms of median and maximum errors. Our method outperforms ASM also in terms of maximum errors in all cases, however AAM is generally better in terms of maximum errors for the first two sets. Newton's method (NM) and gradient ascent (GA) give close results in general. However, Newton optimization converges in



(a) Set 1

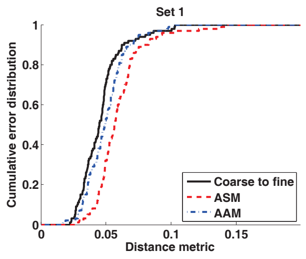


(b) Set 2

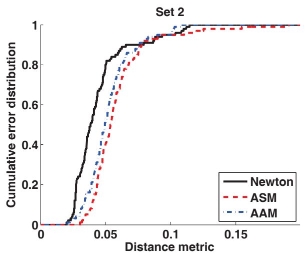


(c) Set 3

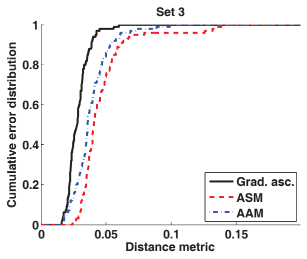
Figure 5: Comparison of cumulative error distributions for different search methods.



(a) Set 1



(b) Set 2



(c) Set 3

Figure 6: Comparison of cumulative error distributions for the best search method of each set with ASM and AAM.

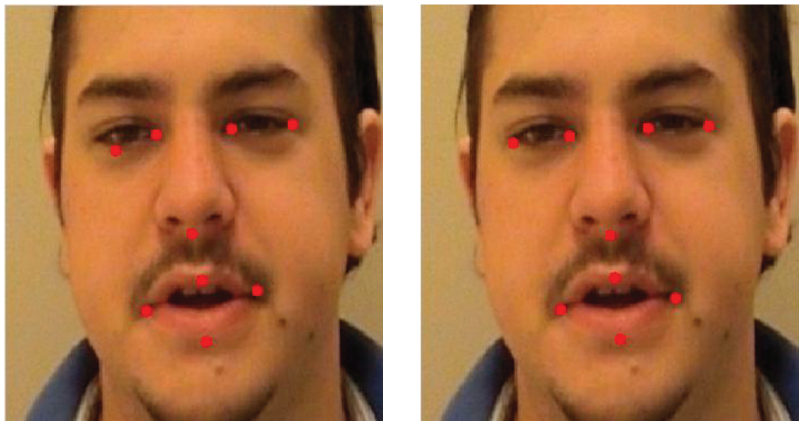

(a) Independent locations (b) Dependent locations

Figure 7: Facial feature locations obtained using independent and dependent locations models, with a good independent model initialization.

fewer iterations, making it faster than gradient ascent optimization. Newton optimization is slightly better than other search methods in terms of median errors, for Set 1 and Set 2 which are more realistic as they have distinct train and test subjects. However for those sets, coarse-to-fine search is better in terms of maximum errors. We attribute this to the incremental nature of this method which gets close to the actual locations in the coarsest resolution and does not allow much deviation from them in finer resolutions, reducing the maximum errors that can be made. Coarse-to-fine search also provides more stable results among different sets. However, training for coarse-to-fine search takes longer, as it needs models for all resolutions that take place during the search. But once it is trained, it allows for a faster search as it needs to perform a full search only in the coarsest resolution. In finer resolutions, the search is very fast since there are few candidate locations. Coordinate ascent results are acceptable and usually run faster than or comparable to the gradient based methods.

Two example facial feature extraction results using the independent model and dependent-locations model with gradient ascent search are shown in Figure 7 and Figure 8.

4.5. Robustness to face detection errors

In this part, we analyze the performance of our method with respect to face detection errors. We performed some experiments where we added

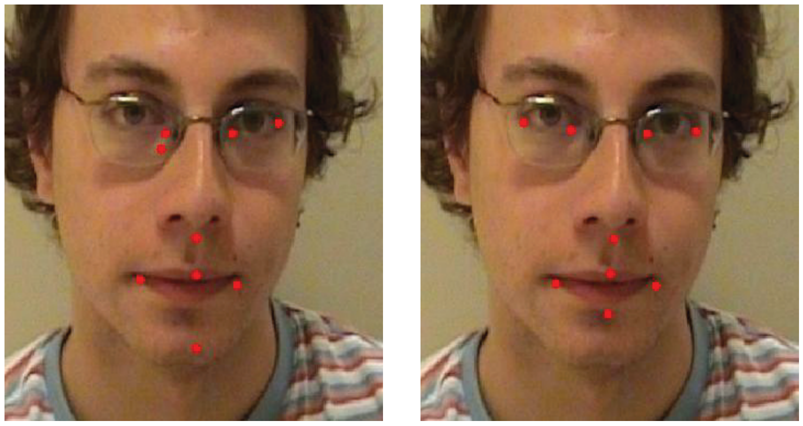

(a) Independent locations (b) Dependent locations

Figure 8: Facial feature locations obtained using independent and dependent location models, with an inaccurate independent model initialization.

noise to regular face detection results. For these experiments, a dataset consisting of IMMDB images is used. First 96 images are used for training and the remaining 120 images are used for testing. We only provide the dependent model results using the Newton's method. Effects of translation errors are given in Figure 9. Here, σ denotes the coefficient of standard deviation of translation errors, which is multiplied by image width and height to obtain a translation error added to the x or y coordinates of both the top-left coordinates and bottom-right coordinates of face detection rectangles quadrupling the effect of the noise. Effectively we move each side of the face detection rectangle using the generated noise. We can see that the performance of the method gets worse with increasing face detection noise as expected. The errors are doubled when there is noise with a standard deviation of 7-8%. However in practice, face detectors are getting better and better and we have not seen such large errors in real cases. Face detectors usually fail totally or detect the face region pretty accurately. We can say that the facial feature localization will work well when the faces are correctly detected or when there is an error in the detected face region less than 5%.

4.6. Image resolution experiments

In this part, we report our experiments investigating the effects of resolution of face images used in detecting facial feature points. We provide the median accuracies and test times for the independent model and dependent

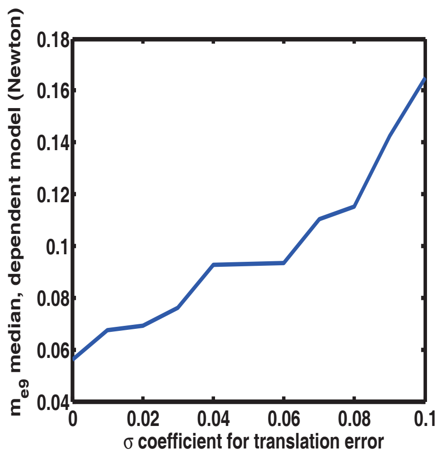


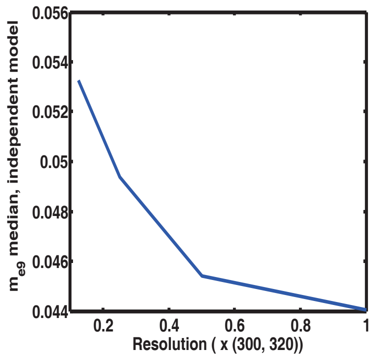
Figure 9: Effects of face detection errors on m_{e9}.

model with Newton's method in Figure 10. Experiments are realized using Set 1. We observe that in the independent model, using a higher resolution image improves the results with a payoff in computation time. However in the dependent model, the accuracies are best when a reduced resolution image (1/4 of the actual dimensions) is used with the added benefit of reduced computation time. We conjecture that in higher resolutions, the gradient search algorithms get stuck in local maxima and yield suboptimal results. For this reason, we use images of quarter resolution to detect facial features using our dependent model.

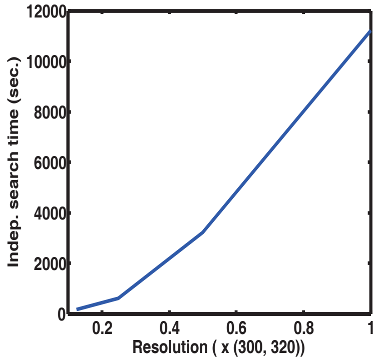
4.7. Experiments with fixed parameters and window sizes

In this part, we analyze the performance of the algorithms when fixed parameters are used for all features. PCA subspaces of dimension 50 are obtained for all facial features by using the texture information inside square patches of fixed size 25×25 around facial feature candidate points in images of resolution 80×75. Also, we apply a single histogram equalization to the grayscale image in these tests. The results are given in Table 4. We observe that the results are comparable to the ones given in Table 2. Thus, we conjecture that our methods are not very sensitive to the specific values of these parameters.

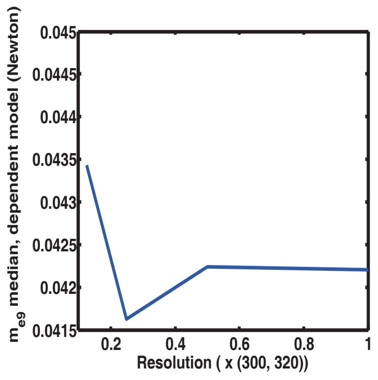
We also report results with double and half the window sizes used in Table 5. The results show that the performance is not exteremely sensitive



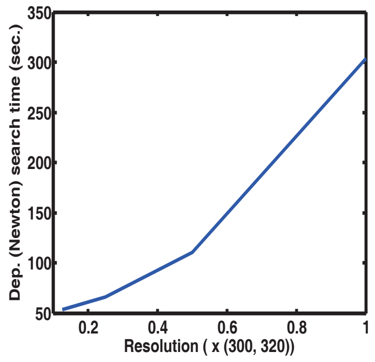
(a) Independent model median error



(b) Independent model test time



(c) Dependent model median error



(d) Dependent model test time

Figure 10: Effects of changing resolutions in median accuracy and test times in the independent and dependent models.

Table 4: Comparison of m_{e9} errors when using fixed parameters.

Set 1	Indep.	Gradient ascent	Newton	Coord. ascent	Coarse-to-fine	AAM
Median	0.0494	0.0419	0.0416	0.0460	0.0426	0.0493
Maximum	0.1449	0.1623	0.1623	0.1462	0.1460	0.1028
Set 2	Indep.	Gradient ascent	Newton	Coord. ascent	Coarse-to-fine	AAM
Median	0.0464	0.0408	0.0403	0.0432	0.0435	0.0494
Maximum	0.1424	0.1432	0.1432	0.1600	0.1466	0.1092
Set 3	Indep.	Gradient ascent	Newton	Coord. ascent	Coarse-to-fine	AAM
Median	0.0348	0.0309	0.0308	0.0337	0.0311	0.0358
Maximum	0.0772	0.0715	0.0847	0.0902	0.0620	0.0990

with respect to the window size used, but the best results are nevertheless obtained with 25×25 windows.

4.8. Computational complexity

Total computation times in seconds for each method that we evaluated in the previous section are given in Table 6 and Table 7 for training and testing on a regular windows PC. We performed the experiments on an Intel Pentium(R) Dual CPU T3400 machine with a 2.16 GHz CPU, 2 GB RAM and 32 bit Windows Vista Home Premium with Service Pack 1 installed on it. Training times are much less important than the testing times. The dependent locations methods require the independent model result for the initialization and score image computation, so their times are indicated with a "+" which indicates additional time. Coarse-to-fine method is an exception to this since it is initialized in the coarsest resolution only. Note that the search algorithms for the dependent locations model are implemented in MATLAB. The AAM and ASM algorithms are realized using C++ implementations available as open source which are AAM-API and STASM respectively. We observe that our methods require about 100 times more run-time as compared to the STASM and 4-5 times more run-time as compared to the AAM-API. However, since our method is implemented in MATLAB, we expect 10-20 times speed improvement when implemented in C++. At least,

Table 5: Comparison of m_{e9} errors for different window sizes.

25×25	Independent	Newton
Median	0.0494	0.0416
Maximum	0.1449	0.1623
13×13	Independent	Newton
Median	0.0563	0.0523
Maximum	0.1528	0.2585
49×49	Independent	Newton
Median	0.0603	0.0553
Maximum	0.1948	0.1895

Table 6: Total training computational time (seconds) for each dataset and method.

Set 1	IND	C2F	ASM	AAM
Time(s)	66.3521	492.1691	26.4	1340.3
Set 2	IND	C2F	ASM	AAM
Time(s)	148.6111	1181.0875	32.5	2128.8
Set 3	IND	C2F	ASM	AAM
Time(s)	310.1110	1719.6017	23.5	2931.2

some common operations such as stacking the pixels of windows around the points to be searched could be compiled as MATLAB mex functions. This alone will speed up the code significantly. In addition, the initial "score image" computation takes the most time and that step is easily parallelizable. We expect even more speed-up by using GPU parallel processing for that step.

We do not argue that we can compete with running times of well-developed and optimized ASM and AAM algorithms in this paper. For this initial study, our goal is to show that we can get better results given enough time and resources. The implementation of our algorithms in MATLAB only serves as a research prototype for a proof of concept. Further optimization and parallelization of the algorithms are planned in the future.

Table 7: Total testing computational time (seconds) for each dataset and method.

Set 1	IND	GA	NM	CA	C2F	ASM	AAM
Time(s)	924.75	+74.80	+103.98	+123.74	1159.91	10.4	424.22

Set 2	IND	GA	NM	CA	C2F	ASM	AAM
Time(s)	1566.87	+118.50	+209.72	+209.19	1284.34	9.5	400.77

Set 3	IND	GA	NM	CA	C2F	ASM	AAM
Time(s)	884.47	+54.73	+96.77	+91.75	879.11	10.1	515.52

5. Conclusions and future work

5.1. Conclusions

It is experimentally shown that our facial feature extraction methods using both the independent model and the dependent-locations model outperform the AAM-API implementation of the AAM method and the STASM implementation of the ASM method for the same experimental setup in terms of normalized errors for most of the cases. Our algorithm is modeling the probability distributions of facial feature locations arising from inter-subject or inter-session differences when there are no major global pose variations. It is critical for our algorithm that it takes as the input, the result of a good face detector.

Dependent-locations model gives better results than independent locations model as expected. Newton optimization usually gives relatively better results than gradient ascent optimization as it additionally takes the second order gradients of the objective function into account. Coordinate ascent using multiple candidates for each feature yields acceptable results but does not perform as good as gradient-based techniques.

Coarse-to-fine search allows fine search around the found locations of coarser resolutions. It yields lower maximum errors than other search methods when the train and test subjects are distinct, also providing a faster search.

If the training data and test data contain similar images in terms of illumination conditions, camera parameters and involve similar or exactly same subjects, then our method works better due to the texture similarity between training and test instances. This way our method can more easily learn the illumination and appearance variations of face images in a database.

5.2. Future work

We were able to get promising facial feature extraction results from independent and dependent-locations assumptions offered in this work. Dependent-locations model improves the independent one. It is in our plans to work with more sophisticated texture parameters in the future. Using global-pose variation compensation is expected to improve our approach as well. Using color information in addition to gray-level intensities can also improve facial feature extraction results. It may be possible to fine tune special parameters experimentally but it would not be fair when comparing our method with general purpose approaches like AAM. It would be an interesting idea to quantify the database dependence of the performance of different methods in more detail. Our method is utilized for point extraction from face images in this work, however it is possible to use it for other purposes, such as medical imaging as well. We envision that other probability distributions for locations and texture can be used in future work. The framework is rich enough to admit many different models as long as efficient search algorithms can be found for them.

6. Acknowledgements

This work has been supported by TUBITAK (Scientific and Technical Research Council of Turkey); research support program (program code 1001), project number 107E015.

7. References

References

[1] D. Reisfeld, Y. Yeshurun, Robust detection of facial features by generalized symmetry, ICPR A (1992) 117–120.

[2] K. Sobottka, I. Pitas, Face localization and facial feature extraction based on shape and color information, ICIP C (1996) 483–486.

[3] E. Saber, A. M. Tekalp, Frontal-view face detection and facial feature extraction using color, shape, and symmetry based cost functions, Pattern Recogn. Lett. 19 (8) (1998) 669–680.

[4] J.-S. Oh, D. W. Kim, J. T. Kim, Y.-I. Yoon, J.-S. Choi, Facial component detection for efficient facial characteristic point extraction, in: ICIAR, 2005, pp. 1125–1132.

[5] S. Jeng, H. Liao, Y. Liu, M. Chern, An efficient approach for facial feature detection using geometrical face model, ICPR C (1998) 426–430.

[6] C. Lin, J. Wu, Automatic facial feature extraction by genetic algorithms, IEEE Trans. Image Process. 8 (6) (1999) 834–845.

[7] M. Zobel, A. Gebhard, D. Paulus, J. Denzler, H. Niemann, Robust facial feature localization by coupled features, in: 4th International Conference on Automatic Face and Gesture Recognition, 2000, pp. 28–30.

[8] L. Zhi-fang, Y. Zhi-sheng, A. Jain, W. Yun-qiong, Face detection and facial feature extraction in color image, in: Computational Intelligence and Multimedia Applications, 2003. ICCIMA 2003. Proceedings. Fifth International Conference on, 2003, pp. 126–130.

[9] A. Gunduz, H. Krim, Facial feature extraction using topological methods, in: Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on, Vol. 1, 2003, pp. I–673–6 vol.1.

[10] E. Bagherian, R. W. Rahmat, N. I. Udzir, Extract of facial feature point, in: IJCSNS, 2009.

[11] S. M. Smith, A new class of corner finder, in: Proc. 3rd British Machine Vision Conference, 1992, pp. 139–148.

[12] C. D. Hua Gu, Guangda Su, Feature points extraction from faces, in: Image and Vision Computing NZ, 2003.

[13] M. Hess, G. Martinez, Facial feature extraction based on the smallest univalue segment assimilating nucleus (susan) algorithm, in: in Proceedings of the Picture Coding Symposium (PCS '04), San Francisco, Calif, USA, 2004.

[14] T. Cootes, C.J.Taylor, Active shape models - smart snakes, in: In British Machine Vision Conference, Springer-Verlag, 1992, pp. 266–275.

[15] T. F. Cootes, C. J. Taylor, D. H. Cooper, J. Graham, Active shape models—their training and application, Comput. Vis. Image Underst. 61 (1) (1995) 38–59.

[16] M. H. Mahoor, M. Abdel-Mottaleb, Facial features extraction in color images using enhanced active shape model, in: FGR '06: Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition, IEEE Computer Society, Washington, DC, USA, 2006, pp. 144–148.

[17] Y. Li, J. H. Lai, P. C. Yuen, Multi-template asm method for feature points detection of facial image with diverse expressions, in: FGR '06: Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition, IEEE Computer Society, Washington, DC, USA, 2006, pp. 435–440.

[18] S. Milborrow, F. Nicolls, Locating facial features with an extended active shape model, in: ECCV '08: Proceedings of the 10th European Conference on Computer Vision, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 504–513.

[19] T. F. Cootes, G. J. Edwards, C. J. Taylor, Active appearance models, Lecture Notes in Computer Science 1407 (1998) 484–498.

[20] T. F. Cootes, G. Edwards, C. Taylor, Comparing active shape models with active appearance models, in: in Proc. British Machine Vision Conf, BMVA Press, 1999, pp. 173–182.

[21] F. Tang, J. Wang, H. Tao, Q. Peng, Probabilistic hierarchical face model for feature localization, in: WACV '07: Proceedings of the Eighth IEEE Workshop on Applications of Computer Vision, IEEE Computer Society, Washington, DC, USA, 2007, p. 53.

[22] R. Niese, A. Al-Hamadi, B. Michaelis, A stereo and color-based method for face pose estimation and facial feature extraction, in: Pattern Recognition, 2006. ICPR 2006. 18th International Conference on, Vol. 1, 2006, pp. 299–302.

[23] R. Gross, I. Matthews, S. Baker, Generic vs. person specific active appearance models, Image Vision Comput. 23 (2005) 1080–1093.

doi:http://dx.doi.org/10.1016/j.imavis.2005.07.009.
URL `http://dx.doi.org/10.1016/j.imavis.2005.07.009`

[24] T. K. Leung, M. Burl, P. Perona, Finding faces in cluttered scenes using random labeled graph matching (1995).

[25] M. Burl, T. K. Leung, P. Perona, Face localization via shape statistics (1995).

[26] K. C. Yow, R. Cipolla, Feature-based human face detection, IMAGE AND VISION COMPUTING 15 (1997) 713–735.

[27] J. Naruniec, W. Skarbek, A. Rama, Face detection and tracking in dynamic background of street., in: S. M. M. de Faria, P. A. A. Assuno (Eds.), SIGMAP, INSTICC Press, 2007, pp. 414–420.

[28] J. Rurainsky, P. Eisert, Eye center localization using adaptive templates, in: in Proc. of CVPR Workshop on Face Processing in Video (FPIV04), Washington DC, 2004.

[29] M. Tipping, C. Bishop, Probabilistic principal component analysis (1997).

[30] `Amazon Mechanical Turk`.
URL `https://www.mturk.com/mturk/welcome`

[31] `LabelMe: The open annotation tool`.
URL `http://labelme.csail.mit.edu/`

[32] M. B. Yilmaz, H. Erdogan, M. Unel, Probabilistic facial feature extraction using joint distribution of location and texture information, in: ISVC '09: Proceedings of the 5th International Symposium on Advances in Visual Computing, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 1171–1180.

[33] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via the em algorithm, Journal of the Royal Statistical Society.

[34] D. C. Liu, J. Nocedal, On the limited memory bfgs method for large scale optimization, Math. Program. 45 (3) (1989) 503–528. doi:http://dx.doi.org/10.1007/BF01589116.

[35] K. Levenberg, A method for the solution of certain non-linear problems in least squares, The Quarterly of Applied Mathematics 2 (1944) 164–168.

[36] D. W. Marquardt, An algorithm for least-squares estimation of nonlinear parameters, SIAM Journal on Applied Mathematics 11 (2) (1963) 431–441.

[37] D. Cristinacce, T. Cootes, Feature detection and tracking with constrained local models, in: 17^{th} British Machine Vision Conference, Edinburgh, UK, 2006, pp. 929–938.

[38] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, Vol. 1, 2001, pp. I–511–I–518 vol.1.

[39] U. Meier, R. Stiefelhagen, J. Yang, A. Waibel, Towards unrestricted lip reading, International Journal of Pattern Recognition and Artificial Intelligence (1999).

[40] `M2VTS Database`.
URL `http://www.tele.ucl.ac.be/PROJECTS/M2VTS/m2fdb.html`

[41] M. B. Stegmann, `IMM Face Database`.
URL `http://www2.imm.dtu.dk/ aam/datasets/datasets.html`

[42] `The AAM-API`.
URL `http://www2.imm.dtu.dk/aam/aamapi/`