Probabilistic Facial Feature Extraction Using Joint Distribution of Location and Texture Information

Mustafa Berkay Yilmaz, Hakan Erdogan, Mustafa Unel

Sabanci University, Faculty of Engineering and Natural Sciences
Istanbul, Turkey
berkayyilmaz@su.sabanciuniv.edu,{haerdogan,munel}@sabanciuniv.edu

Abstract. In this work, we propose a method which can extract critical points on a face using both location and texture information. This new approach can automatically learn feature information from training data. It finds the best facial feature locations by maximizing the joint distribution of location and texture parameters. We first introduce an independence assumption. Then, we improve upon this model by assuming dependence of location parameters but independence of texture parameters. We model combined location parameters with a multivariate Gaussian for computational reasons. The texture parameters are modeled with a Gaussian mixture model. It is shown that the new method outperforms active appearance models for the same experimental setup.

1 Introduction

Modeling flexible shapes is an important problem in vision. Usually, critical points on flexible shapes are detected and then the shape of the object can be deduced from the location of these key points. Face can be considered as a flexible object and critical points on a face can be easily identified. In this paper, we call those 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 face detection, facial expression analysis and lipreading.

Approaches like Active Appearance Models (AAM) and Active Shape Models (ASM) [1] are widely used for the purpose of facial feature extraction. These are very popular methods, however they give favorable results only if the training and test sets consist of a single person. They can not perform as well for person-independent general models.

AAM uses subspaces of location and texture parameters which are learned from training data. However, by default, this learning is not probabilistic and every point in the subspace is considered equally likely[1]. This is highly unrealistic since we believe some configurations in the subspace may have to be favored as compared to other configurations.

[1] In some approaches, distributions of the AAM/ASM coefficients are used as a prior for the model.

In this work, we propose a new probabilistic method which is able to learn both texture and location information of facial features in a person-independent manner. This algorithm expects a face image as the input which is the output of a good face detection algorithm. We show that, using this method, it is possible to find the locations of facial features in a face image with less pixel errors compared to AAM.

The rest of the paper is organized as follows: Section 2 explains our statistical model. Experimental results are presented in Section 3. Finally in Section 4 we conclude the paper and propose some future improvements.

2 Modeling Facial Features

Facial features are critical points in a human face such as lip corners, eye corners and nose tip. Every facial feature is expressed with its location and texture components. Let vector $\boldsymbol{l}_i = [x_i, y_i]^T$ denote the location of the ith feature in a 2D image[2]. $\boldsymbol{t}_i = \boldsymbol{t}_i(\boldsymbol{l}_i)$ is the texture vector 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 paper, 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.

2.1 Independent features model

We can simplify this formula by assuming independence of each feature from each other. Thus, we obtain:

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

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

We can calculate the joint probability $P(\boldsymbol{t}_i, \boldsymbol{l}_i)$ by concatenating texture and location vectors; obtaining a concatenated vector $\boldsymbol{f}_i$ of size $p+2$. We can then assume a parametric distribution for this combined vector 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 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 are not optimal.

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 (realistically) that the textures of each facial feature component is only dependent on its own location and is independent of other locations and other textures. 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 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 Location and texture features

It is possible to use Gabor or SIFT features to model the texture parameters. We preferred a faster alternative for the speed of the algorithm. The texture parameters are extracted from rectangular patches around facial feature points. We train subspace models for them and use p subspace coefficients as representations of textures. Principal component analysis (PCA) is one of the most commonly used subspace models and we used it in this work, as in [2–6]. The location parameters can be represented as x and y coordinates directly.

2.4 Modeling location and texture features

A multivariate Gaussian distribution is defined as follows:

$$\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{N/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{7}$$

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.

For the model defined in 2.1, probability 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:

$$P(\boldsymbol{f}_i) = \sum_{k=1}^{K} w_i^k \mathcal{N}(\boldsymbol{f}_i;\boldsymbol{\mu}_i^k,\boldsymbol{\Sigma}_i^k). \tag{8}$$

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. $\mathcal{N}$ indicates a Gaussian distribution with specified mean vector and covariance matrix.

For the model defined in 2.2, probability $P(\boldsymbol{t}|\boldsymbol{l})$ of texture parameters $\boldsymbol{t}$ given location $\boldsymbol{l}$ is also modeled using a GMM as in equation (8).

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

$$I_i(x,y) = \log\left(P(\boldsymbol{t}_i|\boldsymbol{l}_i = [x\ y]^T)\right). \tag{9}$$

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:

$$P(\boldsymbol{l}) = \mathcal{N}(\boldsymbol{l};\boldsymbol{\mu},\boldsymbol{\Sigma}). \tag{10}$$

Candidate locations for feature i are modeled using a single Gaussian model trained with feature locations from the training database. Marginal Gaussian distribution of location of a feature is thresholded and a binary ellipse region is obtained for that feature. Thus, a search region of ellipse shape is found. GMM scores are calculated inside those ellipses for faster computation.

The model parameters are learned from training data using maximum likelihood. Expectation maximization (EM) algorithm is used to learn the parameters for the GMMs [7].

2.5 Algorithm

For independent features model, we calculate $P(\boldsymbol{f}_i)$ in equation (8) 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 dependent locations model, we propose an algorithm as follows. 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:

$$\phi(\boldsymbol{l}) = \log\left(P(\boldsymbol{t}, \boldsymbol{l})\right) = \log\left(P(\boldsymbol{l})\right) + \sum_{i=1}^{N} \log\left(P(\boldsymbol{t}_i|\boldsymbol{l}_i)\right). \tag{11}$$

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

$$\phi(\boldsymbol{l}) = \frac{-\beta}{2}(\boldsymbol{l}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{l}-\boldsymbol{\mu}) + \sum_{i=1}^{N} I_i(x_i, y_i) + \text{constant}. \tag{12}$$

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 coefficient. $I_i(x, y)$ is the score image of feature i defined in equation (9).

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

$$\hat{\boldsymbol{l}} = \text{argmax}_{\boldsymbol{l}}\, \phi(\boldsymbol{l}). \tag{13}$$

To find this vector, we use the following gradient ascent algorithm:

$$\boldsymbol{l}^{(n)} = \boldsymbol{l}^{(n-1)} + k_n \boldsymbol{\nabla}\phi(\boldsymbol{l}^{(n-1)}). \tag{14}$$

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

$$\boldsymbol{l} = [x_1, y_1, x_2, y_2, ..., x_N, y_N]^T. \tag{15}$$

Then we can find the gradient of ϕ as:

$$\boldsymbol{\nabla}\phi(\boldsymbol{l}) = [\partial\phi/\partial x_1, \partial\phi/\partial y_1, ..., \partial\phi/\partial y_N]^T. \tag{16}$$

For a single feature i:

$$\partial\phi/\partial x_i = \frac{\partial}{\partial x_i}\log P(\boldsymbol{l}) + \sum_{i=1}^{N}\frac{\partial}{\partial x_i}\log P(\boldsymbol{t}_i|\boldsymbol{l}_i). \tag{17}$$

and

$$\partial\phi/\partial y_i = \frac{\partial}{\partial y_i}\log P(\boldsymbol{l}) + \sum_{i=1}^{N}\frac{\partial}{\partial y_i}\log\left(P(\boldsymbol{t}_i|\boldsymbol{l}_i)\right). \tag{18}$$

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:

$$\boldsymbol{l}^{(n)} = \boldsymbol{l}^{(n-1)} + k_n(-\beta\Sigma^{-1}(\boldsymbol{l}^{(n-1)} - \boldsymbol{\mu}) + \mathbf{G}), \tag{19}$$

where

$$\mathbf{G} = \begin{bmatrix} G_x^1(\boldsymbol{l}_1^{n-1}) \\ G_y^1(\boldsymbol{l}_1^{(n-1)}) \\ \ldots \\ G_x^N(\boldsymbol{l}_N^{(n-1)}) \\ G_y^N(\boldsymbol{l}_N^{(n-1)}) \end{bmatrix}. \tag{20}$$

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 coordinate (integers) in the image. $\mathbf{G}$ 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. Since $\boldsymbol{l}^{(n)}$ is a real-valued vector, we use bilinear interpolation to evaluate gradients for non-integer pixel locations. Iterations continue until the location difference between two consecutive iterations is below a stopping criterion.

3 EXPERIMENTAL RESULTS

For training, we used a human face database of 316 images having hand-marked facial feature locations for each image. In this database, texture information inside rectangular patches around the facial features are used to train a PCA subspace model. We used 9 facial features which are left and right eye corners; nose tip; left, right, bottom and upper lip corners. PCA subspaces of different dimensions are obtained by using the texture information inside rectangular patches around these facial features. Also we found mean locations for each feature and the covariance matrix of feature locations. To get rid of side illumination effects in different regions of the image; we applied the 4-region adaptive histogram equalization method in [4] in both the training and testing stages. For a faster

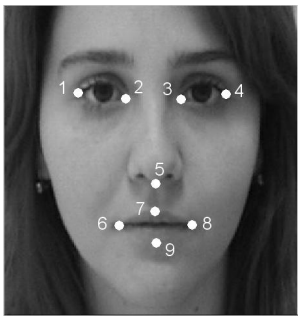


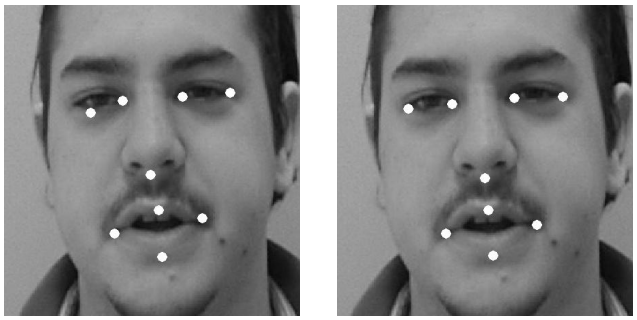
Fig. 1: Facial features used in this work

Table 1: Training parameters used for facial features

Feature	PCA dimension	Window size	Hist. eq.
1	30	8x8	2
2	30	8x8	1
3	30	10x10	2
4	30	5x5	2
5	20	5x5	2
6	50	12x12	1
7	50	10x13	2
8	50	12x12	2
9	50	19x19	2

feature location extraction; training and testing images of size 320x300 are down-sampled to size 80x75, preserving the aspect ratio. Facial features used in our experimental setup are shown in Figure 1. Training parameters used for each facial feature are shown in Table 1. Parameters are: PCA subspace dimension, window size used around facial feature points and histogram equalization method. For 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 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; the values giving the best result is used for each parameter and for each feature. For 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. We used $\beta = 4$ and step size $k_n = 0.05$ for all iterations.

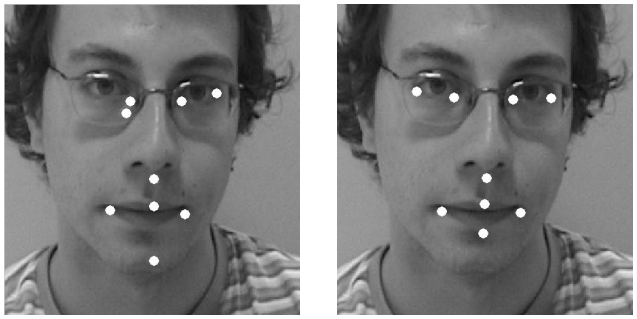
For testing, we used 100 human face images which were not seen before in training data. For the independent model explained in Section 2.1, PCA coefficients and location vectors are concatenated, and a GMM model is used to obtain scores. For each feature, the pixel giving the highest GMM score is selected as the initial location. These locations are then used to solve the dependent locations model in Section 2.2. Using the method explained in Section 2, locations and textures of the features are refined iteratively. Sample results for independent and dependent location models are shown in Figure 2 and in Figure 3. In Figure 3, independent model gives an inaccurate initialization due to limitations of the model. However; dependent locations model corrects the locations of features fairly, using relative positions of features in face. Pixel errors of independent



(a) Independent (b) Dependent locations

Fig. 2: Facial feature locations obtained using independent and dependent locations models, with a good independent model initialization

and dependent locations models using 100 face images of size 320x300 are shown in Table 2. Pixel error of a single facial feature on a face image is the Euclidean distance of the location found for that feature between the manually labeled location of that feature. We find the mean pixel error of all facial features on a



(a) Independent (b) Dependent locations

Fig. 3: Facial feature locations obtained using independent and dependent location models, with an inaccurate independent model initialization

Table 2: Comparison of pixel errors of independent and dependent location models with AAM.

Error	Independent	Dependent	AAM-API
Mean	5.86	4.70	8.13
Maximum	29.55	15.78	17.84

single face image. Mean row in Table 2 denotes the mean of those mean pixel errors over all face images, and maximum row is the maximum pixel error over all face images. Maximum error is shown to indicate the worst case performance for the algorithms.

3.1 Comparison with AAM

Our method is compared with the AAM method [1] using the AAM implementation AAM-API [8]. Note that other AAM search algorithms and implementations such as [9, 10] may perform differently. The same data set is used for training and testing as in Section 3. Comparison of mean and maximum pixel errors with the proposed method is also shown in Table 2. An advantage of AAM is that it takes into account global pose variations. Our algorithm is modeling the probability distributions of facial feature locations arising from inter-subject 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. We are planning to improve our algorithm so that global pose variations will be dealt with.

4 CONCLUSIONS AND 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 dramatically. In addition, the proposed dependent locations model outperforms the AAM. It is in our plans to find better texture parameters. Using global-pose variation compensation is expected to improve our approach.

5 ACKNOWLEDGMENTS

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

References

1. T.F. Cootes, C.T.: Statistical models of appearance for medical image analysis and computer vision. SPIE Medical Imaging (2001)
2. Hasan Demirel, Thomas J. Clarke, P.Y.C.: Adaptive automatic facial feature segmentation. International Conference on Automatic Face and Gesture Recognition (1996)
3. Luettin J, Thacker NA, B.S.: Speaker identification by lipreading. International Conference on Spoken Language Processing (1996)
4. Meier, U., Stiefelhagen, R., Yang, J., Waibel, A.: Towards unrestricted lip reading. International Journal of Pattern Recognition and Artificial Intelligence (1999)
5. P.M. Hillman, J.M. Hannah, P.G.: Global fitting of a facial model to facial features for model-based video coding. International Symposium on Image and Signal Processing and Analysis (2003) 359–364
6. Ozgur, E., Yilmaz, B., Karabalkan, H., Erdogan, H., Unel, M.: Lip segmentation using adaptive color space training. International Conference on Auditory and Visual Speech Processing (2008)
7. A. P. Dempster, N. M. Laird, D.B.R.: Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society (1977)
8. The-AAM-API: (http://www2.imm.dtu.dk/ aam/aamapi/)
9. Matthews, I., Baker, S.: Active appearance models revisited. International Journal of Computer Vision **60** (2003) 135–164
10. Theobald, B.J., Matthews, I., Baker, S.: Evaluating error functions for robust active appearance models. In: Proceedings of the International Conference on Automatic Face and Gesture Recognition. (2006) 149 – 154