Adaptive Automatic Facial Feature Segmentation

Hasan Demirel Thomas J. Clarke Peter Y.K. Cheung
Electrical and Electronic Engineering Department
Imperial College of Science, Technology and Medicine
University of London, London SW7 2BT, U.K.
E-mail: h.demirel@ic.ac.uk

Abstract

Automatic facial feature detection is typically solved by using manually segmented images to train a feature detector. In this paper, we investigate whether it is possible to improve the detection performance of such a feature detector by using additional unsegmented images. We propose a new adaptive automatic facial feature segmentation algorithm which aims to do this. The experimental results using this algorithm demonstrate that it is possible to improve the detection performance obtained from a small segmented training set by using a larger number of additional unsegmented images.

1. Introduction

The identification of facial features, such as eyes, mouth, etc, is important in face recognition, and some of the other facial image processing problems. One important task, *feature detection*, is identifying the position of a facial feature in an image. The feature is then *segmented* by cutting out from the input image the subimage bounding the feature. In general feature segmentation will output the position and size of the identified feature. In this paper we will deal with the simplest type of segmentation where the feature is assumed to be bounded by a fixed size rectangle: in this case feature segmentation can be accomplished with a feature detector.

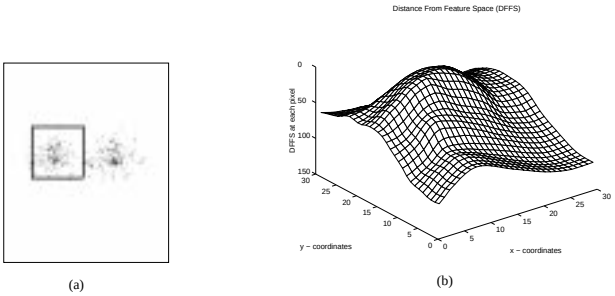
There are two distinct methods of feature detection. One method is to use hand-crafted algorithms, incorporating prior knowledge about the geometry of the feature: for example that an eye pupil is an exact circle. This can achieve good results, but requires complete redesign for every new type of feature to be recognized. The second method, image-based detection, bases recognition on a training set of typical feature images. In this paper we will investigate a novel way of improving image-based detection performance.

Automatic facial feature detectors which use image-based detection have two phases: training and detection. In the training phase, a number of images in which the feature has been segmented - the *initial training set* - are used to train the feature detector. In the detection phase, the detector is given an unsegmented image. It identifies the position of the given feature in the unsegmented image.

In our work, we want to find a way of improving the performance of a feature detector by using additional *unsegmented* images to supplement the initial (segmented) training set. Note that although the location of the desired feature in this additional data is not known, it is known that each unsegmented image contains the feature somewhere. The unsegmented images provide extra information about the desired feature, albeit of a much less reliable nature than that in the initial training set. Unsegmented images (for example from video cameras) are much cheaper to generate than segmented images. In principle the unsegmented image dataset could therefore be thousands of times larger than the initial training set. Under these conditions, the ability to make use of information from unsegmented data could be very valuable.

Let us consider a simple way in which the unsegmented images could be used to augment the initial training set. First use the detector, trained with the initial training set, to segment each of the unsegmented images. Then retrain the detector with a new training set consisting of the initial training set and the newly segmented images. This bootstrapping process is not straightforward. If the image segmentation was incorrect, the extra training data will decrease the accuracy of the detector. If the image segmentation is always correct, a larger training dataset is not needed!

We assume that although the location of the feature in the unsegmented images is not known, its spatial distribution is the same as that in the initial training set. We can therefore use an analysis of the initial training set to determine more and less likely positions of the feature. This assumption is realistic when the unsegmented images are of aligned faces, as is commonly the case. The algorithm we present



(a) (b)

Figure 1. (a) An example of the spatial distribution of right and left eye features. The rectangular region shows the *feature search window* **for the left eye. (b) An example of** *DFFS map* **over the feature search window.**

assigns a weight to each new training image based on the detection confidence of the image and its spatial location. These weights will in general be smaller than the weight given to the initial training set images. In this paper we will demonstrate that by using spatial information in the detection process, and weighting each training image with its reliability, it is possible to obtain significantly better detection performance with the addition of unsegmented images to the training data set than was possible from the initial training set alone.

In our work we use eigenspace coding, derived from Principal Component Analysis (PCA) technique by Turk and Pentland [1]. This technique is briefly explained in section 2. Section 3 below describes the basic automatic feature detection method. Next, in section 4, we will look at how the image weighting is derived. Section 5 describes the adaptive feature segmentation algorithm. Sections 6, 7 and 8 give our methodology, experimental results & discussion and conclusions respectively.

2. Eigenspace Coding

Eigenspace coding is derived from Principal Component Analysis(PCA). PCA is a technique for mapping high dimensional data into a lower dimensional vector space. Assume that the size of training features is $N \times N$ pixels. The training images can be considered as N^2 dimensional vectors. The number of training features is M. By using PCA we obtain M eigenvectors. These eigenvectors are called *eigenfeatures*. Then first $M^{'}$ $(M^{'} \leq M)$ eigenfeatures with highest associated eigenvalues are used to represent each training feature. The linear combination of these eigenfeatures represents the training feature. Hence, input features are mapped into $M^{'}$ dimensional eigenspace. Turk and Pentland give the details of this method in [1].

3. Automatic Facial Feature Detection

There are many approaches to the feature detection problem. Template matching, eigenspace coding [1] have been used, as well as multilayer perceptrons [5] and edge-map projection [6].

The technique of eigenspace coding is commonly used to solve the feature detection problem. It has been shown that eigenspace coding is a more powerful technique than template matching [2], and can tolerate more distortions (e.g. lighting, rotation and scale). The reconstruction error, also called residual error [1], of principal component representation is a very good indicator for a feature detection. The region in eigenspace which contains the cluster of training features is called the *feature space*. In our work we use residual error for feature detection and we call it *Distance From Feature Space(DFFS)* as in [2].

An important pre-processing step in eigenspace coding is to map all the possible pixels of a raw image into eigenspace. In the feature detector this must be done separately for every possible subimage of a raw image which might contain the feature. For each possible pixel, we treat the pixel as the center of a rectangular feature. We then calculate the $DFFS$ for each pixel. We call the resulting 2-D real-valued function the *DFFS map*. We use the *DFFS map* to determine the best possible feature.

A *feature search window* (i.e. rectangular region) is chosen which contains all possible feature positions. In our work we perform feature detection over the feature search window instead of the whole face. This approach reduces the computation time significantly. Figure 1(a) shows the feature search window for the left eye. Figure 1(b) shows an example of a *DFFS map* over a feature search window.

In order to detect the likelihood of a feature having a given position within the feature search window, we use the DFFS map values. The location of the global minimum of the DFFS map determines the most likely position of the feature, given no a priori spatial information.

3.1. Distance From Feature Space

As mentioned in the previous section Distance From Feature Space (DFFS) is the residual description error [1]. It gives the error between the adjusted (a pre-processing step explained below) input image and the reconstructed image by the linear combination of eigenfeatures calculated by using the representation vector and eigenfeatures. DFFS is a clear indication of how close the candidate image is to the feature space of the training set. There are two distance measures we can use for DFFS. These are Euclidian and Mahalanobis distance[3], [4]. Detailed formulation of these two distance measures is given below.

Let Γ_i be the i'th manually segmented training feature

$(i = 1, ..., M)$ where M is the number of training images and the Ψ is the mean feature ($\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i$). u_i $(i = 1, ..., M^{'})$ is the i'th eigenfeature with i'th highest associated eigenvalue, where $M^{'}$ is the number of eigenfeatures used to represent the features. w_i $(i = 1, ..., M^{'})$ is the i'th coefficient of the representation vector. The adjusted input image is $\Phi = \Gamma - \Psi$ and $\Phi_f = \sum_{i=1}^{M^{'}} w_i u_i$. Euclidian distance between the feature and the face space (DFFS) is defined by:

$$\epsilon_e^2 = \| \Phi - \Phi_f \|^2 = \sum_{j=1}^{N}(\Phi_j - \sum_{i=1}^{M^{'}} w_i u_{ij})^2 \qquad (1)$$

N is the number of pixels in the training features. u_{ij} corresponds to the j'th pixel $(j = 1, ..., N)$ of i'th eigenfeature. DFFS using Mahalanobis distance is defined by:

$$\epsilon_m^2 = \sum_{j=1}^{N}(\Phi_j - \sum_{i=1}^{M^{'}} \frac{1}{\sqrt{\lambda_i}} w_i u_{ij})^2 \qquad (2)$$

Here λ_i is the i'th highest eigenvalue which corresponds to i'th eigenvector(eigenfeature).

4. Image Weighting

In the adaptive segmentation phase of the algorithm we automatically segment each new image, and include the segmented feature in the training set. It is necessary to assign a *weight* to each new segmented image to indicate how reliable the segmentation is. We derive from the initial training set two types of information which enable this weight to be calculated. The first, from the feature spatial statistics of the initial training set, gives the probability of the feature being found in a given position. The second, from the DFFS statistics of the initial training set, relates a given DFFS to the probability that the image is the desired feature.

4.1. Feature Spatial Weighting

For the given initial training set we use x and y coordinates of these features over feature search window. The variance in each coordinate (σ_x^2 and σ_x^2) is computed. The covariance between x and y is very small, so we can use a diagonal covariance matrix and calculate a normal probability distribution independently for each coordinate:

$$P(x) = \frac{1}{\sqrt{(2\pi)}\sigma_x} e^{-(x-\mu_x)^2/2\sigma_x^2} \qquad (3)$$

μ_x is the mean in coordinate x. $P(y)$ is computed similarly.

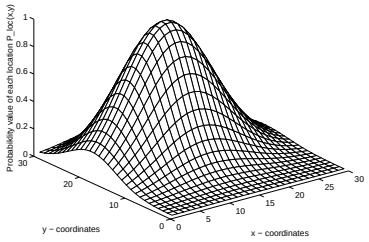


Figure 2. An example of spatial probability distribution of initial training set over feature search window.

Then, for each pixel (x, y) in feature search window the spatial probability distribution is:

$$P_{loc}(x, y) = P(x)P(y) \qquad (4)$$

Figure 2 shows an example of spatial probability distribution of the initial training set over the feature search window. Note that the probability values are rescaled between 0 and 1.

4.2. DFFS Weighting

DFFS weighting is computed by estimating the probability distribution of the distance measures (DFFS) of the initial training set images. For each image in the initial training set we compute the distance between feature space and the image mapped to the eigenspace. We assume that distances have a normal distribution and calculate the corresponding means and variances of $P_{DFFS}(\epsilon)$ for Euclidian and Mahalanobis distance measures. Note that the probability values are rescaled between 0 and 1.

When we get a DFFS map over a feature search window, for each location (x, y) we compute a probability density $P_{DFFS}(\epsilon_{x,y})$, where $\epsilon_{x,y}$ is the DFFS value at location (x, y).

5. Adaptive Feature Segmentation

In *adaptive* feature segmentation we want to have a feature detector which improves its performance in time by adding new features into the training process in an unsupervised manner. To achieve this we propose to add each new feature segmented by the current detector into the training set whenever we are given an unsegmented face image.

We assume that the initial training features are the best features, because they are manually segmented. We assign the highest weights $(weight_i = 1)$ to the features in the initial training set. The images automatically segmented by the adaptive segmentation procedure are not necessarily correct.

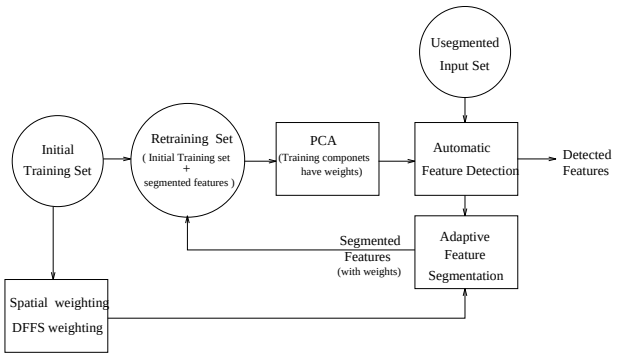


Figure 3. Block diagram of the adaptive automatic facial feature segmentation algorithm.

We assign every such image a weight $(0 \leq weight_i \leq 1)$ depending on the statistics derived from the detection process.

Given a trained feature detector, it is possible to compute the spatial probability density over the feature search window and the DFFS probability of the initial training set as explained in section 4. The two probability measures are independent, so we multiply them to get a weight (proportional to total probability) for each position in the search window. Finally we locate the global maximum (highest weight) over the feature search window. The corresponding position is taken to be the segmented feature. The weight value at this position gives some information about the reliability of this segmentation. The probabilistic arguments in section 4 give this some plausibility, however we make no claim that this choice of weight value is optimal.

5.1. The Algorithm

Here is the complete algorithm:

- *Training*

1. Let Γ_i be the i'th segmented image in the initial training set. Each Γ_i is a rectangular image of the desired feature. Let x_i, y_i be the x and y positions of this image within the corresponding face image. The minimum and maximum values of x_i, y_i determine the size of the feature search window. The parameters of the spatial probability distribution $P_{loc}(x, y)$ over feature search window is estimated from the samples x_i, y_i. (Figure 2).
2. Apply PCA to initial training set to map each feature into feature space. For each image Γ_i in the training set calculate the distance from feature space ϵ_i. Estimate the probability distribution $P_{DFFS}(\epsilon)$ for the set of distances.
3. Assign weight values of 1 to initial training set features Γ_i.
4. Find the mean weighted feature:

$$(\Psi = \frac{1}{\sum_{i=1}^{M} weight_i} \sum_{i=1}^{M} \Gamma_i weight_i) \qquad (5)$$

   Define the adjusted image by $\Phi_i = (\Gamma_i - \Psi) weight_i$. Then apply PCA.

- *Feature detection*

5. Let I be a face image.
6. For each pixel with coordinates (x,y) in the feature search window of I, calculate the DFFS value $\epsilon_{x,y}$. The location (x_d, y_d), which corresponds to the global minimum of DFFS value over the feature search window, is the detected feature.

- *Adaptive segmentation*

7. For each pixel in the feature search window of I, calculate:

$$weight(x, y) = P_{DFFS}(\epsilon_{x,y}) P_{loc}(x, y) \qquad (6)$$

   Find the location (x_s, y_s) for which $weight(x_s, y_s)$ is a global maximum over the feature search window. Let T be the feature at location (x_s, y_s) in I. Add T, x_s, y_s to the training set with $weight(x_s, y_s)$ for adaptive retraining.

8. Go to step 4 to retrain the algorithm.

6. Methodology

In order to test the algorithm we used a database of 400 grey-scale face images from Olivetti Research Laboratory [7]. These images have varied lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images were taken against a

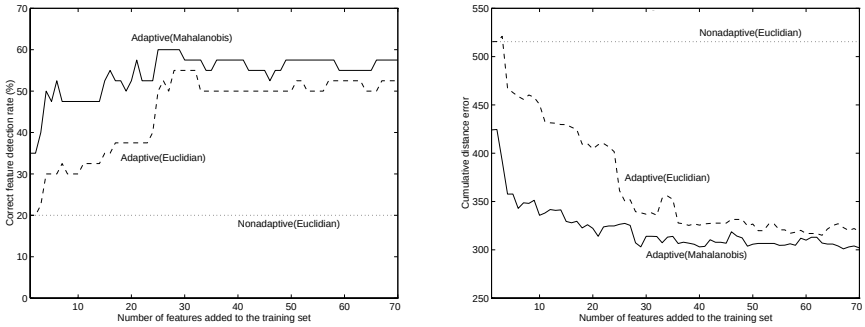


Figure 4. Performance of the feature detector during the adaptation phase. (a) shows the *correct detection rate* **(%), where (b) gives the** *cumulative distance error***, over the unsegmented test set.**

dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement). The size of each image is 92×112 pixels, with 256 grey levels per pixel.

MATLAB and its image processing toolbox was used for implementing the algorithms. The Olivetti database was enhanced by manually segmenting the left eye feature of these face images. The size of the left eye feature was fixed to be 21×25. Figure 5(a) shows examples of some of the manually segmented features.

In our experiments we use three disjoint sets of images from the Olivetti database. A set of 30 database images is used to generate the *initial training set* of left eye features. From each image the 21×25 subimage corresponding to the left eye is extracted as determined by the manual segmentation x_i, y_i. A set of 70 face images without manual segmentation is used as the *input set*. Features segmented automatically from the input set are used to adaptively improve the detection performance of the feature detector. Finally a set of 40 unsegmented face images is used as a *test set*.

Two performance measures are used in our experiments to estimate the performance of the face detector. The *correct detection rate* is the fraction of test set images correctly detected to within a tolerance of 5 pixels (Euclidean distance). The *cumulative distance error* is the sum of individual detection errors (i.e Euclidian distances) in the test set.

7. Results and Discussion

The adaptive segmentation algorithm was tested by the following experiments.

In the first experiment the automatic feature detector is initially trained using the initial training set. It is used to segment a single image from the unsegmented image set. This results in a new segmented image, and a weight. The new image is added to the training set, with the given weight. The feature detector is then retrained using the new training set, and the process repeated. We measure the performance of the feature detector each time we include a new image in the training set. The correct detection rate (%) and cumulative distance error is calculated for the current detection of the test set.

In the second experiment the same data is used, however DFFS is computed with Mahalanobis distance instead of Euclidean distance.

The results of the experiments are shown in Figure 4(a) and (b). In Figure 4(a) the correct detection performance increases as new automatically segmented images are added to the training set. It can be observed that Mahalanobis distance is performing better than compared to Euclidian distance. Figure 4(b) shows a similar improvement in the performance measured by the decrease in the cumulative distance error measure.

In both cases, the results suggest that the adaptive method performs better than nonadaptive method. Figure 5(b) and (c) gives typical correct detections through the adaptive process versus corresponding miss-detection using the initial non-adaptive detector.

The experimental results demonstrate that the detection performance of the feature detector can be increased by using an adaptive segmentation algorithm. For example, when Mahalanobis distance is used, the feature detector trained with only the initial training set has a correct detection rate of %35. After 70 additions of segmented features the correct detection rate increases to %57.5. We measured the change in performance for each addition of a segmented image. You can see that some images decrease the performance. This is expected, because segmentation will not always be correct, and in any case the finite size of the test set makes performance evaluation noisy. However the overall performance clearly increases in time.

(a) (b) (c)

Figure 5. (a) Examples of feature training templates used and, (b) correct detections through the adaptive phase versus (c) typical miss-detections by non-adaptive feature detector.

8. Conclusions

In this paper a novel adaptive facial feature segmentation algorithm, which uses automatic segmentation of unsegmented data to augment an initial, manually segmented, training set, has been presented. The algorithm assigns a weight to each training image which represents its reliability. Our experimental results using this algorithm have demonstrated that it is possible to improve significantly the performance of a facial feature detector, measured in two different ways, by using unsegmented data. We also observed that Mahalanobis distance performs better than Euclidian distance when we use eigenspace coding to perform feature detection.

This novel approach to feature segmentation raises some interesting questions. In principle statistically significant features within an image could be detected by some kind of unsupervised clustering algorithm which required no segmented data. This approach is not usually successful because the lack of segmentation makes the data very noisy. One way of visualizing this is to view a single unsegmented image as a set of n segmented images, one for each possible distinct segmentation. Exactly 1 of these is correctly segmented and gives an example of the correct feature, $n-1$ of these contain noise. In theory, if the noise is uncorrelated, it should be possible to reduce arbitrarily the contribution which this noise makes by having a large enough number of images. However in practice the number required is so large that a small amount of segmented data contains more information than any feasible number of unsegmented images. In our algorithm the noise is filtered by a combination of spatial information and a crude feature detector based on a small initial training set - this makes the unsegmented data relatively more useful.

These initial experiments have given promising results, on a small database. However more work is needed to determine whether the algorithm will have monotonically increasing performance when the number of unsegmented training data images is much larger than the initial training set. If this were so, it would allow accurate feature detectors to be constructed from small training databases of segmented data and very large databases of unsegmented data. The increasing ease with which large quantities of video data can be stored and manipulated makes the prospects for applications in this area particularly exciting.

References

[1] M. Turk and A. Pentland. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3(1):71-86, 1991.

[2] A. Pentland, B. Moghaddam, T. Starner. View-Based and Modular Eigenspaces for Face Recognition. *IEEE Conference on Computer Vision and Pattern Recognition*. 1994

[3] B. Brillault, O. Mahony, T.J.Ellis. A maximum likelihood approach to feature segmentation. *Pattern Recognition*, Vol.26, No. 5, pp. 787-798,1993.

[4] I. Craw, N. Costen, T. Kato, G Robertson and S. Akamatsu. Automatic Face Recognition: Combining Configuration and Texture. *International Workshop on Automatic Face- and Gesture-Recognition*, 1995.

[5] J. M. Vincent, J. B. Waite and D. J. Myers. Automatic Location of Visual Features by a System of Multilayered Perceptrons. *IEE Proceedings* 139(6), Dec. 1992.

[6] T. Kanade. Picture processing by computer complex and recognition of human faces. *Tech. Report, Kyoto University, Dept. of Information Science*, 1973.

[7] Olivetti Research Laboratory face database. *http://www.cam-orl.co.uk/facedatabase.html*