Precise Detailed Detection of Faces and Facial Features

Liya Ding and Aleix M. Martinez
Dept. Electrical and Computer Engineering
The Ohio State University, Columbus, OH 43210
{dingl, aleix}@ece.osu.edu

Abstract

Face detection has advanced dramatically over the past three decades. Algorithms can now quite reliably detect faces in clutter in or near real time. However, much still needs to be done to provide an accurate and detailed description of external and internal features. This paper presents an approach to achieve this goal. Previous learning algorithms have had limited success on this task because the shape and texture of facial features varies widely under changing expression, pose and illumination. We address this problem with the use of subclass divisions. In this approach, we use an algorithm to automatically divide the training samples of each facial feature into a set of subclasses, each representing a distinct construction of the same facial component (e.g., close versus open eye lids). The key idea used to achieve accurate detections is to not only learn the textural information of the facial feature to be detected but that of its context (i.e., surroundings). This process permits a precise detection of key facial features. We then combine this approach with edge and color segmentation to provide an accurate and detailed detection of the shape of the major facial features (brows, eyes, nose, mouth and chin). We use this face detection algorithm to obtain precise descriptions of the facial features in video sequences of American Sign Language (ASL) sentences, where the variability in expressions can be extreme. Extensive experimental validation demonstrates our method is almost as precise as manual detection, $\sim 2\%$ error.

1. Introduction

Face detection has received considerably attention during the past several years [16]. By face detection we generally mean that a bounding box (or an ellipsoid) enclosing the face (or faces) in an image at approximately the correct scale needs be specified. Several of these methods include an estimate detection of the eyes and mouth centers [10], with a few detectors able to estimate the position of several other features such as eye and mouth corners [5, 15]. This additional information is essential in many applications as, for example, in the warping of the texture map of frontal faces before these can be correctly used within the appearance-based framework [7].

In many other applications, however, this face detection process does not suffice and a very accurate and detailed detection of the shape of the major facial features, as that shown in Fig. 1, is needed. This is needed if we are to do recognition of faces using shape information [1, 2] or to correctly warp a face to its shape-free representation. Accurate and detailed facial feature detection is also necessary in the analysis of facial expressions with FACS (Facial Action Coding System) [4, 11] and in the analysis of non-manuals in American Sign Language (ASL) [9]. ASL non-manuals are facial expressions that convey grammatical information in a signed sentence. Another important application of accurate and precise face detection is in the construction of active shape models [2] and shape analysis [6], which require of labeled data during training.

In the present paper, we derive an approach that achieves the precise face detection of facial features illustrated in the first row in Fig. 1. The proposed approach has an average accuracy of 6.2 pixels, which is around 2% in terms of the size of the face. This error is compared with a manual detection. For manual detection we asked several people to manually delineate the contours of the same facial components detected by our algorithm. Exact instructions on how to do the marking were given to each participant. One such detection is shown in the second row in Fig. 1. The variability (or error) among these manual markings was around 4.2 (or $\sim 1.5\%$), only slightly better than that of our algorithm. One can thus conclude that the detection obtained with the proposed algorithm is *detailed* and *accurate*.

We still talk of face (and facial feature) detection, rather than shape modeling, because the algorithms developed herein correspond to this category. Furthermore, our algorithm does not make use of any learned model as in [2, 12]. Instead, our algorithm is based on the generic training of classifiers robust to shape and texture variations due to expression, pose and illumination changes. The training of such classifiers has proven difficult in the past. This is

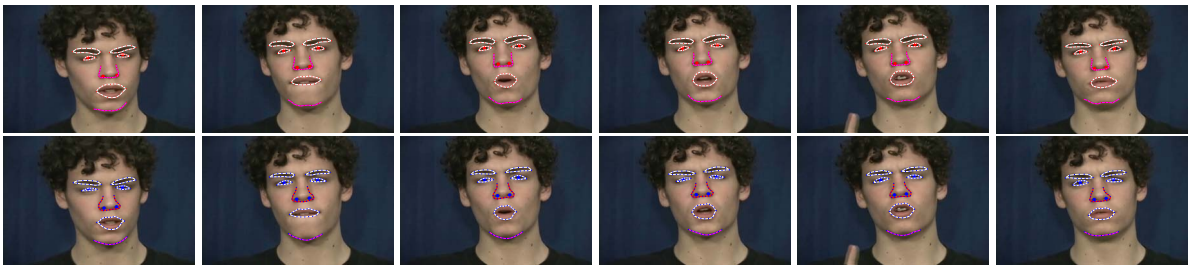


Figure 1. Shown here is an example of accurate and detailed face detection in each of the frames over a video sequences of an ASL sentence. The top row shows the automatic detection obtained with the algorithm defined in this paper. The bottom row provides a manual detection for comparison.

due to the nature of each facial feature. For example, an eye can be open or close, and when open the iris may be looking at different directions. Add illumination changes to this generative process, and the task of detecting eyes becomes extremely challenging. Indeed several algorithms have been designed to specifically address this single issue [10]. Here, we take a different view. First, to detect each facial feature (herein referred as class) under varying generative condition, we employ an algorithm that learns to divide the data into an appropriate number of subclasses. Each of these subclasses will represent a distinct image condition that cannot be learned in conjunction with the others when using the learning methods of choice. This process facilitates the detection even under extreme conditions, yet may not guarantee accurate detection. To provide precise detections, we learn to discriminate between the surroundings of each facial feature from itself. This prevents detections in areas around the actual feature and improves accuracy. Once a large number of such classifiers provide a predetection of the facial features, we employ edge detection and color to find the connecting path between them. This results in the final detection shown in Fig. 1. We provide experimental results on the detection of these facial features over video sequences of ASL sentences (such as that in the figure) where the expression, pose, and illumination over the facial features vary substantially. Our videos also include some partial occlusions, but we do not consider rotations beyond 25^o in each direction since this would result in large self-occlusions.

2. Face Detection in Video

Before we can do facial feature detection, we need to find where the faces are in the image. Many algorithms already exist for this. A fast and reliable algorithm for this purpose is the cascade-based face detection algorithm of Viola and Jones [14]. A face detection example obtained with this algorithm is given in the top row in Fig. 2. As any other detector, Viola and Jones algorithm will sometimes detect the same face twice at different scales or will detect faces where there are none. These two cases are depicted in the examples shown in the first row in Fig. 2. Errors such as these can be readily detected in video sequences using a Gaussian model.

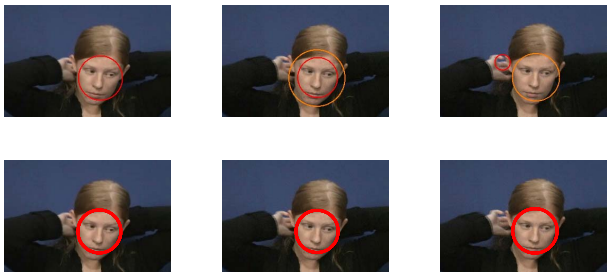


Figure 2. 1st row: Original detection of three consecutive frames using the algorithm of [14]. 2nd row: Corrected results using a Gaussian model.

Let $\mathbf{f}_t = (x, y, s)^T$, where (x, y) is the detected face center, s the scale, and t the frame number. We now fit a Gaussian model over the face detection in all frames, $N(\mu_\mathbf{f}, \mathbf{\Sigma_f})$, with $\mu_\mathbf{f} = (\mu_x, \mu_y, \mu_s)^T$ the mean and $\mathbf{\Sigma_f} = diag(\sigma_x, \sigma_y, \sigma_s)$ the variances. If several faces are detected in each image, they can be described as $\mathbf{f}_{ti} = (x_{ti}, y_{ti}, r_{ti})^T$, where the maximum of i may vary as t increases. The Mahalanobis distance,

$$(\mathbf{f}_{ti} - \mu_\mathbf{f})^T \mathbf{\Sigma}_\mathbf{f}^{-1} (\mathbf{f}_{ti} - \mu_\mathbf{f})$$

provides a simple measure to detect outliers. In our experiments, we consider distances larger than 1 to correspond to false detections. This value has been found to be adequate on the training images.

The process described in the preceding paragraph will provide the number of faces detected over the video sequence. If a frame has one or more of these faces missing (false negative), this will be directly specified by the different values that i takes at that frame. The location of the missed faces can be estimated using the same Gaussian model presented above, since this will provide a smooth transition between frames. It is worth mentioning here that we only considered a face has been missed when the number of frames where the maximum of i is smaller than their neighbors is 2. This allows faces to appear and disappear

at any time, although this is not the case in the video sequences we will use in this paper. An example of this process is shown in Fig. 2, where the bottom row shows the corrected detection over the same three frames shown in the top row.

3. Eyes Detection

After (holistic) face detection, eyes are the facial feature to have received the largest attention, mostly because these play a major role in face recognition and human-computer interaction problems. Unfortunately, accurate detection of the eye centers has been difficult to achieve, in many occasions being required to resort to highly sophisticated methods to achieve this goal [10].

A major reason behind the difficulty of precise eye centner detection is the high variability of these. Although most eyes may seem quite the same at first, closer analysis of a set of cropped eyes (i.e., in isolation) revels a different picture. Eyes may have very distinct shapes (mostly across ethnicity and race, but not exclusively), pupil size, and colors. Furthermore, cast shadows, glasses, and lighting have a strong influence on how eyes are seen in an image. On top of it all, eyes can be open, close or any way in between, and the iris may be pointing at any direction. For now we are interested in finding the center of the eye, regardless of the position of the iris, as well as the bounding box of the eye region (similar to that used to delineate the face). This can be considered as a first crude detection of the eye.

The common approach to building a classifier that learns this high variability of eyes has met with difficulties, even when powerful non-linear algorithms and thousands of training examples have been used. A second problem is that the resulting classifier may be able to give an estimate over the position of the eye's location, but fail to provide an accurate detection of its center. To resolve these issues without resorting to complex algorithms, we will use the idea of Subclass Discriminant Analysis (SDA) [17]. In the 2-class classification problem, the goal of SDA is to learn to discriminate between the samples in the two classes by dividing these into a set of subclasses. The algorithm starts with a single subclass per class. If this is not sufficient to successfully separate the training sample in the two specified classes, the algorithm divides each class into two subclasses. This process is repeated until the training samples are correctly classified or until a stability criterion of the generalized eigenvalue decomposition equation is attained [17]. Here, we use this main idea but provide a slightly different formulation. In our approach, each class is divided into a number of subclasses using the well-known K-means clustering algorithm because this permits to model the many non-linearities of the training data efficiently. As to how many subclasses, the stability criterion of [8] is utilized, which (in our experiments) provided a value of $K = 15$. This addresses the first of the issues mentioned at the beginning of this paragraph. The second problem, that of providing a precise detection of the eye center, can now be nicely addressed using the same formulation. Note that we are yet to define what the two classes of our algorithm are. One is the eye itself, and we can use thousands of images of eyes to train the classifier. The other class is *key* for the accurate detection of the first class. This is defined next.

We note that when a classifier does not precisely detect the facial feature it has been trained on, it usually provides a close estimate. This is because cast shadows, eyeglasses and others will make a neighboring area look more like an eye to the classifier. To resolve this, we can train the classifier to discriminate between eyes and their neighboring areas. This will create a pulling effect toward the desirable location. Similar ideas have been successfully used for (holistic) face detection [13]. For eye detection, the first class will be the eyes, represented by 941 images of cropped eyes, while the second class will correspond to copped images of the same size (24×30 pixels) located in the neighboring areas of the eye. Fig. 3 shows how the eye window is centered (class 1) while the eight accompanying background windows (class 2) are located off center. This yields a total of $7,528$ background images. To increase robustness to scale and rotation, we expanded the training set by including the images obtained when re-scaling the original training images from .9 to 1.1 at intervals of .1, and by adding random rotated versions of them within the range of $-15°$ to $15°$. Images are mean- and norm-normalized to make them invariant to the intensity of the light source. We also trained two independent classifiers, one for the left eye and another for the right. In each case, we obtained a subspace representation, $\mathbf{V}$, where the discrimination between the two classes is maximized [17].

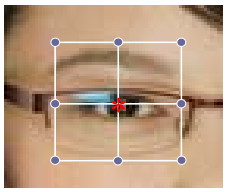


Figure 3. The red star in the figure corresponds to the center for the eye window, used to generate the training data for eyes. The blue dots represent the window centers for the background samples. The distance from the eye center and the background window is set to 24 pixels.

To detect the eyes in a new image, we do an exhaustive search in a pre-specified region within the face detection box. This potential eye regions were obtained from a statistical analysis of the eye location over the manually detected eyes previously used to train our classifiers. The goal is to establish where the eyes are with respect to the bounding box found by the holistic face detector used. These regions are shown in Fig. 4. In actuality, in the figure, we show two eye regions per eye. To detect the eye centers in

a previously unseen face, we first search for them within the smaller green region (since this includes $\sim 90\%$ of the eye centers in our training set). If the eye's centers are not successfully located in this region, we move our search to the wider blue region (which includes 100% of the training cases). These regions are in effect the priors of our classifier, and although one could also estimate the distribution within them, a simple uniform probability provides the results we need. Note that these search regions are not only tuned to the classifiers previously defined, they also make the search much faster and robust.

As in training, the test image is also re-scaled at $s = \{1.1, 1, 0.9\}$. At each scale, each of the cropped images $\mathbf{x}_{sj}$, of 24×30 pixels and centered at the j^{th} pixel within the eye-search region, is compared to the learned subclasses,

$$\arg\min_i (\mathbf{V}^T\mathbf{x}_{sj} - \mathbf{V}^T\mu_i)^T\mathbf{V}^T\mathbf{\Sigma}_i^{-1}\mathbf{V}(\mathbf{V}^T\mathbf{x}_{sj} - \mathbf{V}^T\mu_i),$$

where $(\mu_i, \mathbf{\Sigma}_i)$ are the mean and covariance matrix of the i^{th} subclass, with subclasses 1 through K representing the first class (eye) and subclasses $K+1$ to $2K$ the second class (background). As mentioned earlier, subclass discrimination analysis is computed in a subspace defined by the projection matrix $\mathbf{V}$ [17]. The minimum of these distances gives the class label of the j^{th} position at scale s within the eye region shown in Fig. 4(a). Let us denote this class label as $D(u_{sj}, v_{sj}, s)$, where $(u_{sj}, v_{sj})^T$ is the center opossition of the window image $\mathbf{x}_{sj}$, and s is the scale. The results obtained at different scales are normalized and added, $D(u_j, v_j, 1) = \sum_s D(u_{sj}/s, v_{sj}/s, s/s)$. This provides a voting over each location. Two examples are shown in Fig. 4(b). Detected eye regions having a small overlap with the top voted region are reclassified as background. The final estimation of the eye position (center and bounding box) is given by the mean of all resulting overlaps, Fig. 4(c).

If a video sequence is available, as it is the case in our application, we can further refine the detection as we did in Section 2 for faces. Here, we use a similar Gaussian model to that employed earlier. The only difference is that this modeling includes the positions of the two eyes as well as the angle θ defined between the horizontal and the line connecting the two eye centers. Detected outliers (i.e., false detections) are eliminated and substituted by an interpolation between the previous and subsequent frames with correct detection.

4. Detecting the Eye Shape

With the face and eyes detected, we can move to the next phase of extracting the detailed information we need. The very first thing we need to do is to provide the left and right margins for each eye. This we can do if we successfully detect the eye corners. To achieve this, we repeat the exact same process defined in Section 3 for detecting the eye centers but apply it to the detection of corners. The same

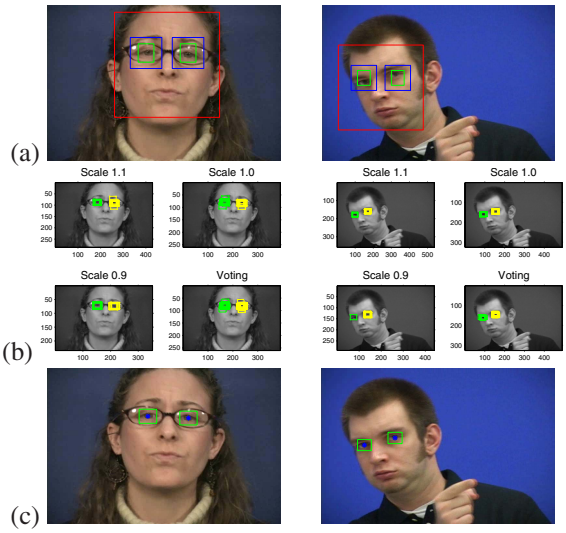


Figure 4. (a) Priors: region where the eye centers are in the training images. (b) Voting: results of the detection of the eye centers at different scales and positions. (c) Final detection of the eye region and center.

process is needed here because eye corners also conform to a large variety of shapes, textures and colors (makeup and eyeglasses being a major problem that needs to be learned). We build two detectors: one to detect inner corners and another for the outer. We train on the left eye and apply it to detect both. To detect the corners of the right eye we simply flipped the image (i.e., mirror image). An eye corner detection is shown in Fig. 5.

The next step is to delineate the iris region. In grayscale, the iris is almost always the darkest area within the eye detected region (regardless of eye color). The iris is thus readily detected as the minimum of all the average circle areas, which can be defined as a convolution,

$$\mathbf{ImP} = \text{conv}(\mathbf{Im}, \mathbf{H})$$

$$(u_p, v_p) = \arg\min_{u,v} \mathbf{ImP}(u, v),$$

where $\mathbf{Im}$ is the normalized grayscale images and $\mathbf{H}$ is a circle mask of radius r_I. The iris radius r_I is learned (independently) for each subject as the one that gives the highest average gradient in $\mathbf{ImP}$. This method could have false detection if the image included heavy shadows or dark makeup around the eye area. To prevent these false detections, we first obtain all local minima and then select the one that has the largest gradient between each detected local minimum and the eye. The highest gradient will be given when moving from the darkness of the iris to the whiteness of the conjunctiva, making it a robust detector. Fig. 5 shows the detected iris as a circle.

While the iris region is dark, the eye lids are of skin color. This is especially true for the lower lid, since this has a very limited mobility and is not highly affected by deformations,

shadows and others. However, the lids need closer analysis becuase makeup and small occlusions (such as eyeglasses) may cause problems. To address this, we test the correlation of the lid with various line orientations θ. The one that gives the highest average gradient in its normal direction is chosen as the best match. Having the eye corners and points for each of the lids, the contours can be build by fitting a cubic spline through the detected points. The final result is illustrated in Fig. 5.

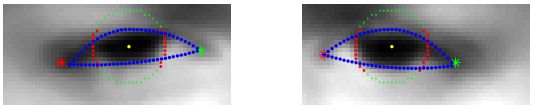


Figure 5. Eye corners are represented with an asterisk. The iris is shown as a circle, of which, the red part is the visible component. The upper and lower contours of the shape of the eye are shown in blue.

5. Detection of Other Facial Features

The approach defined above can now be used to detect the remaining of the facial features. Moreover, with a good detection of the face and the eyes, the location of the rest of features is already approximately known. The easiest of these are the nose and the eyebrows. For example, the x position of the eyebrows is very restrictive, while their y position cannot vary much. Since the eyebrows are either darker or lighter than the skin, it is easy to detect these by searching for non-skin color in the region above the eyes.

To model skin color, we use a Gaussian model defined in the HSV color space, $N(\mu_c, \mathbf{\Sigma}_c)$, with $\mu_c = (\mu_H, \mu_S, \mu_V)^T$ and $\mathbf{\Sigma}_c = diag(\sigma_H, \sigma_S, \sigma_V)$. Over 3 million skin-color sample points were used to train this model. The pixels above the eye regions that do not fit to the color model $N(\mu_c, \mathbf{\Sigma}_c)$ are eliminated first. A Laplacian operator is then applied to the grayscale image above the eyes. From the remaining of the pixels, the ones with highest gradient in each column are kept as potential descriptors of the eyebrows. A binary image morphology is applied to the result so as to obtain a uniform region. Only the largest, continuous region is kept. Two results are shown in Fig. 6.

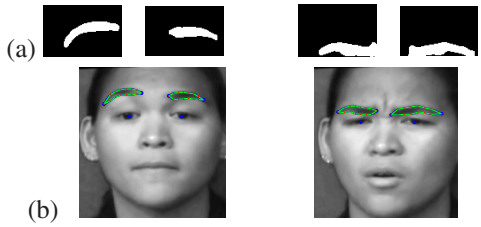


Figure 6. Two examples of eyebrow detection. (a) Binary description of the brow region. (b) Final contur detection.

The position of the nose is arguably the easiest to estimate, because contrary to other features such as the eyes, brows and mouth, the nose cannot undergo large deformations. However, extracting the contour of the nose is very challenging, since this is highly influenced by cast shadows or unclear texture. Cast shadows are especially strong for caucasians, who have larger noses. Smoothness of texture is more typical of Asian faces, making it difficult to specify where the nose ends (even during manual detection). What we do know is that the nose should be within the two eyes about the x axis and below these about the y axis. The nose region is thus defined as that below the lower eye lid and between the two eye centers.

We train a nose classifier following the exact same procedure detailed in Section 3. This provides a bounding box describing the location of the nose, Fig. 7(a). To extract the nose contour, we calculate the gradient of the image and generate its projection onto the x and y axes, Fig. 7(b). This gives us two histograms of the shape, G_x and G_y. To eliminate outliers, such as shadows and makeup, we find the bounding box containing $max(G_x)/2$ and $max(G_y)/2$. This provides a tighter, more precise estimate of the location of the nose. The nostrils are detected as the two darkest (i.e., two minima in graylevel) points within this tighter bounding box, Fig. 7(b). The outer gradient curve of this new bounding box is considered the nose edge. To improve accuracy, we also impose the nose curve to move around the pixels with lowest graylevel value within this detected gradient region/trajectory. The final result is shown in Fig. 7(b).

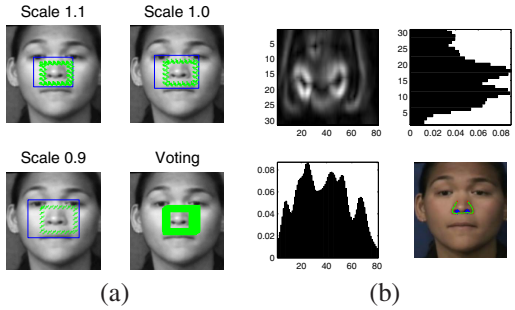
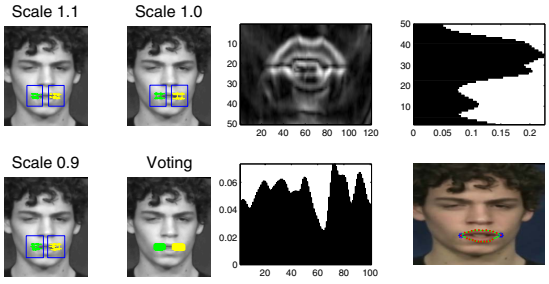


Figure 7. (a) Shown here is an example of nose detection using the subclass classifier defined in this paper. Refinement is done with a voting approach over various scales. (b) From left to right and top to bottom: gradient of the nose region found in (a), y projection of the gradient, x projection, and final detection of the nose shape and nostrils.

The mouth is highly deformable, making an accurate and precise detection challenging. Sophisticated algorithms have been recently defined to address this problem [3]. Here, we use our methodology to derive a fast, accurate and detailed detection without the need to resort to complex or time consuming tasks. To this end, a mouth corner detector is defined using the subclass-based classifier presented in Section 3. We only train a left mouth classifier. To detect the right corners, we create mirror images of the testing windows. An example is given in Fig. 8(a). The bounding

box of the mouth is obtained following the same procedure described above for the nose, Fig. 8(b). Mouths are rich in color, which makes the final process of delineating them feasible. Here, we use a similar process to that employed earlier: skin color and Laplacian edge detection. In particular, we extract three features, given by saturation, hue, and Laplacian edges. Each of these masks are threshold at values T_s, T_h and T_g before being combine into a single mask response. T_s is set as the average saturation value in our color model. T_h is set as the mean hue minus one third of the standard deviation. T_g is the mean of the gradient. An example result is shown in Fig. 8(b). The final extraction of the contour of the upper and lower lips are given in by the outer contour of the mouth mask, Fig. 8(b).



(a) (b)

Figure 8. (a) Mouth corner detection. (b) Gradient of the mouth window, its x and y projection, and the final result.

To conclude, we only need to delineate the bottom limit of the face, i.e., the chin. The chin is generally given by a slightly curve edge below the mouth. However, this can be easily occluded by cast shadows or, in ASL, by the hand. To address this issue, we first extract the edges from a bounding box located immediately below the mouth and then find the best match between the edge points and a quadratic (ellipsoidal-like) curve. The shortest distances from the points to the best fit, d, are calculated and assigned positive or negative labels depending on whether these are above or beneath the current best fit. A Gaussian model is fitted to this result, $N(\mu_d, \sigma_d)$. The edge points with distance $|d|$ larger than T_d pixels are eliminated, with T_d being the larger one of 10 and $2\sigma_d$. The fitting process is repeated over the remaining points. The process is reiterated until no points are excluded. The remaining of the points corresponds to the detection of the chin. Examples of the final fit were given in Fig. 1. Three examples with occlusions are now shown in Fig. 9.

Figure 9. A few examples of chin detection with partial occlusions.

6. Experimental Results

As promised, we now illustrate the uses of the proposed approach on a set of video sequences of ASL sentences, where the facial expression plays a major role. We used ten video sequences of approximately 76 frames, each sequence corresponding to one of 10 different ASL signers. This provides us with a total of 766 frames. Each face is enclosed in a window of approximately 300×300 pixels. We want to see how our face detection compares to that of manual detection.

To properly compare our results with those obtained with manual detection, we need to have a representation that is common to both. For this, we first used the method defined in this paper to find the shape (contours) for each of the internal facial features and the chin. We then resample these curves with a total of 98 equally distanced feature points. The corner points specify the beginning and end of a curve and facilitate comparison with a manual marking result. To be robust to temporary occlusions and noise, a Kalman filter is utilized, which takes into account the motion continuity in the video sequence. An example was shown in Fig. 1. We now show five additional examples in Fig. 10.

To perform a fair comparison with a manual detection result, we provided three "judges" with specific instructions on how to mark 139 feature points around the same facial features detected by our algorithm. The judges had the option of magnifying the images at each location to facilitate accurate detections. After marking each image, the facial feature contours were obtained with a least-squares fit. Here too, each resulting curve was resampled with a total of 98 feature points following the same procedure defined in the preceding paragraph. A result is shown in the second row in Fig. 1.

The process of resampling the manual and automatic detections with the same facial features provides a simple mechanism to calculate their difference. A comparison over the 700 frames used gives a mean error (i.e., the difference between the two) of 6.23 pixels and a standard deviation of 1.52 pixels. In terms of face size, we can estimate the percentage of error as $E = \frac{e}{2r}$, where e is the error in pixels and r the radius of the face (also in pixels). Using this equation, the percentage of error of our automatic estimate is 2.1%.

To determine how accurate this result is, we want to know what is the average difference between manual markings. That is, when different judges do the marking, the final result varies. We want to know what is the usual (acceptable) variability. A comparison among judges resulted in an average difference of 4.2 pixels with a standard deviation of 1 pixel. This corresponds to a percentage of error of 1.5%, which is only slightly better to that given by our automatic detection. We can hence conclude that the proposed algorithm provides a very good estimate on the video sequences used here.

Figure 10. Shown here are five examples of the automatic detection of faces and facial features as given by the proposed approach.

7. Conclusion

Considerable advances have been recently registered in the area of face detection. However, few address the issue of detailed and accurate facial feature detection. In this paper we have presented an approach to remedy this shortcoming. Our method is based on learning to discriminate between features and their surroundings and on a voting strategy over different scales. Facial features are then delineated using gradient and color information. Results on 700 frames show that the proposed algorithm is comparable to manual marking.

Acknowledgments

This research was partially supported by NSF grant 0713055 and NIH grant R01 DC 005241.

References

[1] R. Brunelli and T. Poggio, "Face recognition: features versus templates," PAMI 15:1042-1053, 1993. 1

[2] T.F. Cootes, C.J. Taylor, D.H. Cooper and J Graham, "Active shape models – their training and application," CVIU 61:38-59, 1995. 1

[3] F. De la Torre, J. Campoy, Z. Ambadar and J.F. Cohn, "Temporal Segmentation of Facial Behavior," ICCV, 2007. 5

[4] P. Ekman and W.V. Friesen, "The Facial Action Coding System: A Technique For The Measurement of Facial Movement," Consulting Psychologists Press, 1978. 1

[5] B. Heisele, T. Serre, M. Pontil and T. Poogio "Component-based Face Detection," CVPR, pp. I.657 -662, 2001. 1

[6] I.L. Dryden and K.V. Mardia, "Statistical Shape Analysis," John Wiley, 1998. 1

[7] A.M. Martinez, "Recognizing Imprecisely Localized, Partially Occluded and Expression Variant Faces from a Single Sample per Class," PAMI 24:748-763, 2002. 1

[8] A.M. Martinez and M. Zhu, "Where are linear feature extraction methods applicable?," PAMI 27:1934-1944, 2005. 3

[9] M.S. Messing and R. Campbell, "Gesture, Speech, and Sign," Oxford University Press, 1999. 1

[10] T. Moriyama, T. Kanade, J. Xiao and J.F. Cohn, "Meticulously Detailed Eye Region Model and Its Application to Analysis of Facial Images," PAMI 28:738-752, 2006. 1, 2, 3

[11] M. Pantic and L.J.M. Rothkrantz, "Automatic analysis of facial expressions: the state of the art," PAMI 22:1424-1445, 2000. 1

[12] S. Romdhani and T. Vetter, "3D Probabilistic Feature Point Model for Object Detection and Recognition," CVPR, 2007. 1

[13] K. Sung and T. Poggio, "Example-based learning for view-based human face detection," PAMI 20:39-51, 1998. 3

[14] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," CVPR, pp. I. 511-518, 2001. 2

[15] D. Vukadinovic and M. Pantic, "Fully automatic facial feature point detection using Gabor feature based boosted classifiers," SMC, pp.1692-1698, 2005. 1

[16] M. Yang, D.J. Kriegman, and N. Ahuja, "Detecting Faces in Images: A Survey," PAMI 24:34-58, 2002. 1

[17] M. Zhu and A.M. Martinez, "Subclass Discriminant Analysis," PAMI 28:1274-1286, 2006. 3, 4