Facial Feature Tracking, Extraction and Selection

Summer Internship Report
Meysam Shahrbaf(B.Sc. student) and Daniel Khashabi(B.Sc. student)

Under the supervision of
Mr. Mahdi Kalayeh(M.Sc.), Illinois Institute of Technology
Dr. Hamid Sheikhzadeh(Ph.D.), Amirkabir University of Technology

Media Processing Laboratory
Amirkabir University of Technology, 2011

Contents

Abstract

In this report, first we explain how we moved toward implementing an algorithm for tracking facial features. The explanations, thoroughly covers our experiences, both our failures and successful trials. As a failed experienced of facial feature tracking, we have explained the correlation-based tracking and its improved extension. The second tracking method, is devoted to Active Appearance Model and its details. The implementation of the explained model showed that the AAM model has enough ability for real-time facial feature tracking. Simultaneously, we followed extraction of facial features using Gabor wavelet and selecting facial feature points algorithms. Eventually, due to high dimension of extracted feature vectors derived by the use of Gabor wavelet, we moved to selecting the most elite sup-population of the features which will result in the most discriminative classification procedure and have lowest redundant information. Hence, Adaboost method and PCA based algorithms are exploited so as to reduce the dimension of the generated feature vector. The comprehensive explanation of the feature selection methods has been brought in the main context.

1 Introduction

Our summer internship is the preliminary part of our overall B.Sc. thesis project. The final goal is to control a computer using facial commands, i.e. sequence of facial expressions. In this case, we make discrimination between *facial commands* with sister terms *facial expressions* and *facial gestures*. To make more clear, we call *command*, as a sequence of facial gestures during a specific period of time. The ability to recognize such a facial command recognition system, we should develop a system that track the states of face as times passes. As a first step in reaching the facial commands, we need to track the facial feature on any face and derive the information on the face. Most of the time of the summer internship, was devoted to the employment of several feature tracking algorithms, e.g. Active Appearance Model and generation of facial feature by the use of Gabor wavelet and the selection of facial feature using Adaboost and PCA methods. At the current time we are planning to apply the generated feature to multi-kernel Relevance Vector Machine classifier combined with a Markov model of the face states, for prediction and generation of facial commands, the results of which will be published within a paper.

2 Facial Feature Tracking

Tracking the features laid on face is one of the most important part of this project. At the first look, facial features' tracking appears so important because they are needed for localizing the estimated position of decisive parts on face, e.g. estimated position of eyes, cheeks, lip, etc; moreover, tracking decisive points on face can be used to extract any arbitrary distance-based features on face, e.g. distance between upper and lower lips, wideness of eyes, etc.
Following assumptions are made in the tracking implementation:

- The person's face is almost in direct and stable condition relative to the camera. This is because the camera is laid on the pupil's head. Thus the algorithm should only be sensitive to slight head motions.
- However the camera is fixed on the head, the algorithm must be robust enough to camera's gradual sliding caused by factors like perspiration or person's motion.
- Appropriate transformation is applied to eliminate poor illumination in the picture; thus, details of the face are fairly distinguishable.

2.1 Correlation-based facial feature tracking

2.1.1 Basic model

Based on the method used in [3], one can track any point moving in two consecutive frames of a video, by the use of *correlation* operation. As it is shown in Figure 1, by the use of a little square centred in the feature point in the ith frame, assuming that the feature has a definite maximum velocity which does not let it to move more than a limited distance between two specific frames, we look for a shape inside the

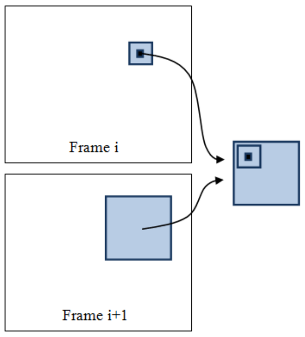


Figure 1: Correlation between two consecutive frames in with one big square(new frame) and little square(old frame).

big square of $(i+1)$th frame, similar to the little square of the ith frame, which can simply be done by calculating correlation between the big and the little square. The point with the maximum correlation is the new position of the feature, in frame $i+1$. The algorithm of the aforementioned method is elucidated in the Algorithm 1.
There is a direct relation between the size of the squares and the maximum speed of the features that the method can track; the more big we assume the squares(especially bigger one), the more fast motions we can track. On the other hand, increasing size of the squares causes the algorithm to run in significantly slow rate. In additions, the more we increase the size of the squares the accuracy of the tracking deteriorates.
To classify the flaw with this method, we can say:

- The tracking has a few error in every cycle, whose accumulation in consecutive cycles causes the features to go far off from the tracking points; due to limited movement of features on face, even a little deviation of tracking points can cause considerable amount of error in the output.
- Sometimes the accidental similarity of the points can cause, erroneous displacement of the target tracking point; as a noticeable example, one can refer to the similar color skin in of the various parts of the face. This matter causes the algorithm to be unstable.

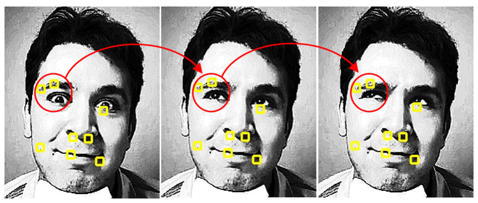
Algorithm 1 Correlation-based feature tracking

```
Define $bigSquareSize$, $smallSquareSize$
loop
  Form the small square $S_i = crop(frame_i, smallSquareSize, centeredAt: featurePoint)$
  Form the big square $S_{i+1} = crop(frame_{i+1}, bigSquareSize, centeredAt: featurePoint)$
  Calculate the correlation between the $S_i$ and $S_{i+1}$: $c(x,y) = \sum_j \sum_k S_i(j,k) S_{i+1}(x+j, y+k)$
  Find the position of the maximum correlation: $ind_{max} = \arg\max_{x,y}\ c(x,y)$
  Correlation offset: $corrOffset = ind_{max} - size(S_i)$
  Rectangle offset: $rectangleOffset = bigSquareSize - smallSquareSize$
  Calculate the overall offset: $offset = rectangleOffset + corrOffset$
  Update the feature point: $featurePoint \longleftarrow featurePoint + offset$
end loop
```

2.1.2 Hierarchical correlation tracking

Based on experiments done with the method explained with the previous method, following results derived:

- The more small square we choose, the more precisely it can track the feature points. But the flimsy point of such case is its inability in tracking sudden fast movement of the object. In sum, smaller squares are rather precise trackers, but not robust against fast movements.


(a) The points are tracked almost correctly.

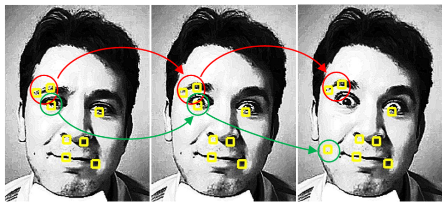

(b) Some points are tracked correctly(red) and some incorrectly jumped to another similar point(green).

Figure 2: Correlation-based facial feature tracking in 3 consecutive frames.

- The bigger squares are more robust in tracking sudden movements. But such big squares have significant robustness in tracking the fast movements, due to embracing a vast number pixels.

The two above points give us the idea of hierarchical employment of both large and small squares which is elucidated in the Algorithm 2. If we assume a set of squares for the correlation process, as in the algorithm, the position of the feature is found by sequential employment of the Algorithm 1 starting with the largest square and to the smallest square. In fact, by employing a big set of squares, the approximate position of the feature begins. The smaller squares have the role of finding the exact position of the feature on face, where the approximate position of the feature is previously determined by bigger squares.
In sum, after implementation of the Algorithm 2, the tracking ability of the algorithm, both in the sense of tracking accuracy and tracking robustness at the presence of sudden movements significantly increased; the noticeable flaw of the new hierarchical method is its huge amount computational process demand, which makes it an impractical methods. Yet, increasing the accuracy of tracking by this method, the algorithm still lacks morphological positioning of the features on face and their relative positioning. This gives us the ability to track some points that solely based on their color. We should admit that there are some untrackable points due to their nondescript position; but it is possible to approximate their position on face, with considering relative movements of their neighbouring points. The imperfections mentioned before, made us to move toward a state-of-the-art algorithm of Active Appearance Model.

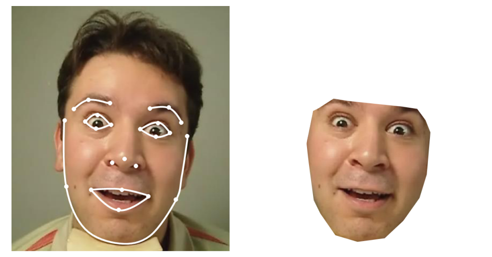
Algorithm 2 Hierarchical correlation for feature tracking

```
Define set of **K** big and small squares $bigSquareSize_k$, $smallSquareSize_k$
**loop**
  Define the estimated feature point $estimateFeaturePoint = featurePoint$
  **for** $k = 1$ to **K** **do**
    Form the small square $S_i = crop(frame_i, smallSquareSize_k, centeredAt: \ featurePoint)$
    Form the big square $S_{i+1} = crop(frame_{i+1}, bigSquareSize_k, centeredAt: \ estimatedFeaturePoint)$

    Calculate the correlation of the $S_i$ and $S_{i+1}$: $c(x,y) = \sum_i \sum_j S_i(x,y) S_{i+1}(x+i, y+j)$
    Find the position of the maximum correlation: $ind_{max} = \arg\max_{x,y} \ c(x,y)$
    Correlation offset: $corrOffset = ind_{max} - size(S_i)$
    Rectangle offset: $rectangleOffset = bigSquareSize - smallSquareSize$
    Calculate the overall offset: $offset = rectangleOffset + corrOffset$
    Update the feature point: $estimatedFeaturePoint \longleftarrow estimatedFeaturePoint + offset$
  **end for**
  Update the feature point: $featurePoint \longleftarrow estimatedFeaturePoint$
**end loop**
```

2.2 Active Appearance Model

Regarding the deformable state of human face, a desirable algorithm should consider specific characteristics inherent in human face. In fact, when tracking several feature points on face, it is vital to consider the



(a) ShapeModels shown with white lines. (b) Appearance (texture) model of the face area.

Figure 3: Sample appearance and shape values

correlative movement of the features on face, i.e. no two point can take any two relative state to each other. Thus the model should consider the state of the features and their deformation in relation to each other. The AAM method explained in [4]. As it is explained in the aforementioned reference, the AAM consists of a linear model face. As it is explained, we can decompose characteristic of face into two group of facial *appearances* and facial *shape*. As it is depicted in the Figure 3(a), the *shape* model consists global positioning of the features on face; by changing the facial expression, laughing, crying, ... one distorts the default positioning of the facial feature points. At what follows, it will be shown that, however the basic structure of the *appreance* and *shape* is linear, the procedures of training and fitting are non-linear[4].
The AAM is first proposed in [5, 6, 7, 8]. However due to shape-appearance structure of the of the AAM, it is applicable to any deformable object, but it is mostly used for tracking of facial feature point tracking in a sequence of images, i.e. video [9, 10, 11]. It is important to mention that the AAM is just one member of a broad set of shape and appearance algorithms,i.e. Active Shape Model(ASM), Direct Appearance Model(ADM), Active Blobs and Morphable Models.
One can separate the AAM algorithm into two online and offline time. In the offline time we aim at training the algorithm based on manually warped images, the input data-set; thus, similar to conventions in[4], we name this section *training phase*. The online time consists of fitting the model on the localized image of the face(object) and finding best deformation of the image that fits the output image, with the least absolute sum of the error; we name this phase the *fitting phase*.
It is noteworthy to mention that here it is assumed that the input images are localized to fit only the small amount of the background image. To state the matter in other way, in a real application of the AAM, it is essential to find the proximity of the deformed object using another algorithm, e.g. in the case of facial feature tracking, a face recognition algorithm can be exploited so as to find the approximate place of face and then we can apply on that approximate position. Another point in the input images is that, they should all normalized in the sense of the translation, scale and rotation. A pre-eminent approach toward this goal can be achieved by Procrustes method as in [12]; therefore, the AAM is only dealing with some local shape and appearance deformations.

2.2.1 Shape model

As it is mentioned before the *shape* model is a set of positions, that make up the mesh of the shape model. To define the *shape* model based on common terminology [4]:

$$\mathbf{s} = \mathbf{s}_0 + \sum_{i=1}^{n} p_i \mathbf{s}_i$$

$$\mathbf{s} = [x_1, y_1, x_2, y_2, ..., x_v, y_v]$$

Regarding the above formula, the shape model is *linear* by considering the mean shape of the input data-set features points, $\mathbf{s}_0$ and the constitutional shape vectors $\{\mathbf{s}_i\}_{i=1}^{n}$ to be at hand. It should be elucidated that the set of $\{\mathbf{s}_i\}_{i=1}^{n}$ consists of possible displacements of facial features, but the mean shape vector, $\mathbf{s}_0$ is

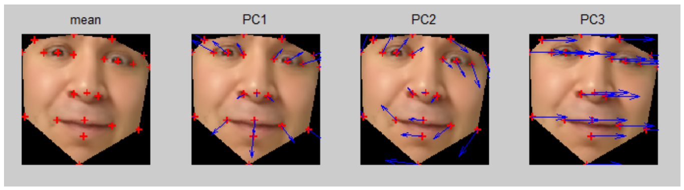


Figure 4: Three first principle components of the shape model.

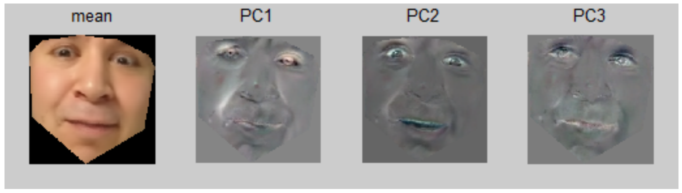


Figure 5: Three first principle components of the appearance model.

roughly the position of features on face. To decipher in another way, we can express the linear shape formula as deviations of the mean shape vector $\mathbf{s}_0$ to the extent of the $\sum_{i=1}^{n} p_i\mathbf{s}_i$. The set of $\{p_i\}_{i=1}^{n}$ is called shape parameters.
The constitutional shape vectors $\{\mathbf{s}_i\}_{i=1}^{n}$ should calculated in a away that make as sparse and spanning as possible space for the input training data. One can simply use the well-known rank reduction method of Principal Component Analysis(PCA) for this aim. As it is mentioned in [8], the principle shape components are those eigenvectors corresponding to the the biggest set of the eigenvalues in the correlation matrix(covariance matrix). Based on the spanning characteristic of the principle shape components derived from the training data, it must has been revealed that the expressiveness of the input images is highly important from the sense of the generating an expressive shapes vectors $\{\mathbf{s}_i\}_{i=1}^{n}$.

2.2.2 Appearance model

Like the shape mode, the *appearance* model is also a linear function of constitutional appearance functions $A_i(\mathbf{x})$. In order to make the comparison of two appearances of in two images, it is reasonable to first make them have same shape; in other words, in defining the appearance model, it is considered that the face(object) has the mean shape $\mathbf{s}_0$, i.e. are *shapeless*. Thus, we define $\mathbf{x} = (x, y) \in \mathbf{s}_0$, as [4]:

$$A(\mathbf{x}) = A_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x}), \qquad \forall \mathbf{x} \in \mathbf{s}_0$$

in which $A(\mathbf{x})$ shows the appearance(texture, intensity) over the all pixels of the face(object). In fact, the appearance model, yields in a picture, with the shape of $\mathbf{s}_0$(shapeless), whose intensities are deviated from the mean intensities, $A_0(\mathbf{x})$ to the extent of $\sum_{i=1}^{m} \lambda_i A_i(\mathbf{x})$. Comparing to the shape model, the mean appearance, $A_0(\mathbf{x})$ is average appearance of the input data and the sequence $\{\lambda_i\}_{i=1}^{m}$ the appearance parameters, is possible appearance deviations from the mean appearance.
Similar to the shape model, the $A(\mathbf{x})$ are selected by the help of PCA dimension reduction; choosing eigenvectors with the maximum corresponding eigenvalues, which yield to the most sparse and the most accurate approximation of the target image, as it is explained in [4].

2.2.3 Shape deformation

As it is mentioned all the images in the training procedure of the appearance model, should have the mean shape, $\mathbf{s}_0$(shapeless). At the time of training, because all the training inputs are manually warped, it is possible to deform all of them to the shape of as mean shape, $\mathbf{s}_0$. Such deformation can be done by a bunch of ways, e.g. triangulating the warped points and by definition of a piecewise affine warp between corresponding triangles[8]. To make it clear, first, the corresponding triangles are found and then by creating

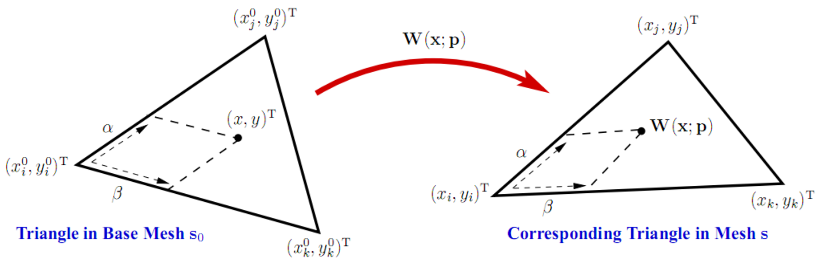


Figure 6: bi-linear deformation between two corresponding triangles. Image from [13].

the warp function between two triangles, values from the deformed image are bi-linearly sampled. To be more exact, if we define the warp function $\mathbf{W}(\mathbf{x};\mathbf{p})$ with parameters $\mathbf{p}$, the $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$ is the deformed image based on the warp function $\mathbf{W}$. Based the Figure 6, by considering the distance values α and β, for each point in the first triangle, we can compute its correspond point in the second triangle [4]. The method is also called *Piecewise Affine Warp*. Such a warp function can defined as the following:

$$\mathbf{W}(\mathbf{x},\mathbf{p}) = (x_i, y_i)^T + \alpha\left[(x_j, y_j)^T - (x_i, y_i)^T\right] + \left[(x_k, y_k)^T - (x_i, y_i)^T\right]$$

in which the values are depicted in the Figure 6. In fact, by this way, we do the inverse warp of image I back to the mean-shape, $\mathbf{s}_0$. The values α and β only depend the input pixel and selected triangles. In overall, one can write the warp function as:

$$\mathbf{W}(\mathbf{x};\mathbf{p}) = (a_1 + a_2 x + a_3 y, a_4 + a_5 x + a_6 y)^T \tag{1}$$

To state in another way, one can find the warp function between to corresponding triangles by calculating only six constants $\{a_i\}_{i=1}^6$. The procedure for piecewise affine method is elucidated in the Algorithm 3:

Algorithm 3 Piecewise affine deforming

```
for all triangles in the mean-shape $s_0$ do
    Compute the corresponding vertex coordinates using the shape mode.
    Compute the warp function $\{a_i\}_{i=1}^6$
    For each pixel in the mean-shape triangle, calculate the value of $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$
end for
```

Till here, we have only defined independent; in the *combined AAM* model[8], the appearance and shape parameters are one-by-one coupled, based on rationale that the appearance and shape variations are mutually dependent. In this case a third PCA should be employed on the combined AAM model for deriving a more compact model[8]; where in the *independent AAM* model, the appearance and shape parameters vary independently. The same as [4] we exploit the independent model of the AAM, due to its simplicity and applicability.

2.2.4 Model fitting procedure

The fitting procedure, consists of minimizing the error between the input image and the approximated output image of the model. Such fitting problem is a nonlinear optimization problem. If we define the error for each pixel of an image, the error image[4]:

$$E(\mathbf{x}) = A_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x}) - I(W(\mathbf{x};\mathbf{p}))$$

where the $\mathbf{W}(\mathbf{x};\mathbf{p})$ is the corresponding pixel of $\mathbf{x}$ under the shape deformation with parameters $\{p_i\}_{i=1}^n$. Thus $I(W(\mathbf{x};\mathbf{p}))$ is the corresponding intensity to the pixel $\mathbf{x}$ in the input image. The goal is to minimize the cumulative error of the fitting:

$$E_{cum} = \sum_{\mathbf{x}\in\mathbf{s}_0} \left[A_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x}) - I(W(\mathbf{x};\mathbf{p})) \right]$$

Regarding the aforementioned error function, the appearance models should be calculated within a $\mathbf{s}_0$ shape, at the time of analyzing any given picture, the algorithm, at the first step, should calculate the deformed shape of the target image, and change it to mean shape (shapeless form), as an input image of the appearance model; and at the rest, fitting the linear model of the appearance to the shapeless input image. Thus, in a AAM fitting problem, the minimization should be done simultaneously with respect to both shape parameter $\{p_i\}_{i=1}^{n}$ and appearance parameters $\{\lambda_i\}_{i=1}^{m}$. For the sake of convenience, temporarily we omit the appearance parameters and only work with $\{p_i\}_{i=1}^{n}$.
Three general methods for fitting problem is explained in [14], namely *Forward Additive*, *Forward Compositional* and *Inverse Compositional*. More exact analysis of the methods fitting methods for the general image alignment problems can be found in [13]. Such problem can be so challenging because of the nonlinear nature of such optimization; In addition to non-linearity, huge number of pixels is another hindrance in fast reaching to the solution, because of the need for calculating lots of gradients on the image pixels. The goal is to approximate or pre-compute the gradient values to reach to a fast method of fitting.
The Lucas-Kanade alignment [14] is a method of additive updating the image parameters. In fact, if we only consider a target image, T($\mathbf{x}$); and an input image, I($\mathbf{x}$) and the the warp function $\mathbf{W}(\mathbf{x};\mathbf{p})$ with parameters $\mathbf{p}$, we aim at minimizing $\sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x})]^2$ with additive update $p \leftarrow p + \Delta p$:

$$E_{\mathbf{p}+\Delta\mathbf{p}} = \sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) - T(\mathbf{x})]^2$$

Rewriting the Taylor expansion of the above formula:

$$E_{\mathbf{p}+\Delta\mathbf{p}} = \sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} - T(\mathbf{x}) \right]^2$$

$$\frac{\partial}{\partial \Delta\mathbf{p}} E_{\mathbf{p}+\Delta\mathbf{p}} = \sum_{\mathbf{x}} 2 \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} - T(\mathbf{x}) \right] = 0$$

$$\Delta\mathbf{p} = H^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$$

where H, the Hessian matrix is:

$$H = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]$$

In sum, the procedures for the Lucas-Kanade algorithm is shown in the Algorithm 4. One can simply calculate the computational complexity of the algorithm $O(n^3)$, where the largest computational demand is lied in inverting the Hessian matrix, H. Due to high computational demand and slowness of Lucas-Kanade, it is merely impossible to use this method in real applications.

The *Inverse Compositional Method* postulates that the $\mathbf{\Delta p}$ in the square error summation can be seen in a different way; as it is suggested in the [14], we can enter the $\mathbf{\Delta p}$ by another warping function into the target image:

$$E_{\mathbf{p},\Delta\mathbf{p}} = \sum_{\mathbf{x}} [T(\mathbf{W}(\mathbf{x};\Delta\mathbf{p})) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]^2$$

where we can write the Taylor expansion for the $T(\mathbf{W}(\mathbf{x};\Delta\mathbf{p}))$ as:

$$E_{\mathbf{p},\Delta\mathbf{p}} = \sum_{\mathbf{x}} \left[T(\mathbf{W}(\mathbf{x};0)) + \nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]^2$$

Algorithm 4 Lucas-Kanade Image Alignment

```
while $|\Delta\mathbf{p}| \leq \epsilon$ do
    Calculate $I(\mathbf{W}(\mathbf{x;p}))$, the warped image
    Compute the error $E = T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x;p}))$
    Warp the gradient $\nabla I$ with $\mathbf{W}(\mathbf{x;p})$
    Evaluate the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $(\mathbf{x;p})$
    Compute the steepest descent images $\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
    Compute Hessian matrix $\mathbf{H} = \sum_x \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]$
    Compute $\sum_x \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x;p}))]$
    Compute $\Delta\mathbf{p} = \mathbf{H}^{-1} \sum_x \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x;p}))]$
    Update the parameter $\mathbf{p} \longleftarrow \mathbf{p} + \Delta\mathbf{p}$
end while
```

By differentiating the above expression, we have:

$$\frac{\partial}{\partial \Delta\mathbf{p}} E_{\mathbf{p},\Delta\mathbf{p}} = \sum_{\mathbf{x}} 2 \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T \left[T(\mathbf{W}(\mathbf{x};0)) + \nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} - I(\mathbf{W}(\mathbf{x};\mathbf{p}))\right] = 0$$

therefore,

$$\Delta\mathbf{p} = -\sum_{\mathbf{x}} H^{-1} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right] [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$$

and the Hessian matrix is:

$$H = \sum_{\mathbf{x}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]$$

The main difference of the *Inverse Compositional* method with the *Lucas-Kanade* method is emanated from calculation of $\left.\frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right|_{\mathbf{W}(\mathbf{x};0)} = \mathbf{cte}$ in the *Inverse Compositional* method and $\left.\frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right|_{\mathbf{W}(\mathbf{x};\mathbf{p})}$ in the *Lucas-Kanade* which is changing in every iteration. Thus, in the *Inverse Compositional* method it is sufficient to pre-calculate the constant values of $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ and H^{-1} once at the beginning of the algorithm.
In the [14], it is shown that two fitting methods *Inverse Compositional* and *Lucas-Kanade* are equivalent. The *Inverse Compositional* fitting algorithm is shown in the Algorithm 5. An intermediary algorithm called *Forward Compositional* is also mentioned in the [14], which we have withheld to explain. More empirical surveys on fitting algorithms can be found in [13].

The updating Δp in every cycle applied by the update $\mathbf{W}(\mathbf{x;p}) \longleftarrow \mathbf{W}(\mathbf{x;p}) \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{p})^{-1}$, where we need to calculate $\mathbf{W}(\mathbf{x};\Delta\mathbf{p})^{-1}$. Due to non-linearity of the warp function, we cannot easily calculate its inverse. As in the[14], we can use an approximation of $\mathbf{W}(\mathbf{x};\Delta\mathbf{p})^{-1}$ instead of its exact value:

$$\mathbf{W}(\mathbf{x};\Delta\mathbf{p}) = \mathbf{W}(\mathbf{x};0) + \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} = \mathbf{x} + \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p}$$

$$\mathbf{W}(\mathbf{x};-\Delta\mathbf{p}) = \mathbf{W}(\mathbf{x};0) - \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} = \mathbf{x} - \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p}$$

By composing two warp functions we have :

$$\mathbf{W}(\mathbf{x};\Delta\mathbf{p}) \circ \mathbf{W}(\mathbf{x};-\Delta\mathbf{p}) = \mathbf{x} + \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} - \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} + O(\Delta\mathbf{p}^2) \simeq \mathbf{x}$$

$$\mathbf{W}(\mathbf{x};-\Delta\mathbf{p}) = \mathbf{W}(\mathbf{x};\Delta\mathbf{p})^{-1} \tag{2}$$

Algorithm 5 Inverse Compositional Algorithm

```
Compute $\nabla T$
Evaluate $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $(\mathbf{x};\mathbf{0})$
Compute the steepest descent image $\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
Compute Hessian matrix $\mathbf{H} = \sum_x \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]$
while $|\Delta \mathbf{p}| \leq \epsilon$ do
    Calculate $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$, the warped image
    Compute the error $E = I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x})$
    Compute $\sum_x \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T [I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x})]$
    Compute $\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_x \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T [I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x})]$
    Update the warp $\mathbf{W}(\mathbf{x};\mathbf{p}) \longleftarrow \mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta \mathbf{p})^{-1}$
end while
```

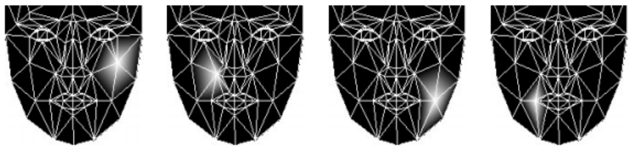


Figure 7: Warp Jacobians with respect to the vertices' positions. Image from [15].

For calculation of the Jacobian, one should use the chain rule:

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \sum_{i=1}^{n} \left[\frac{\partial \mathbf{W}}{\partial x_i} \frac{\partial x_i}{\partial \mathbf{p}} + \frac{\partial \mathbf{W}}{\partial y_i} \frac{\partial y_i}{\partial \mathbf{p}} \right]$$

where (x_i, y_i) is vertex set of the shape model and n, as mentioned before, number of shape models. We divide the computation of the Jacobian into two steps[4]: (1) $\frac{\partial \mathbf{W}}{\partial x_i}$ and $\frac{\partial \mathbf{W}}{\partial y_i}$, (2) $\frac{\partial x_i}{\partial \mathbf{p}}$ and $\frac{\partial y_i}{\partial \mathbf{p}}$. The $\frac{\partial \mathbf{W}}{\partial y_i}$ is the rate of changes in the warp function with respect to the changes of the vertices, which can be straightly derived by the warp generated function in equation 1, as follows:

$$\frac{\partial \mathbf{W}}{\partial x_i} = (1 - \alpha - \beta, 0)^T \quad \frac{\partial \mathbf{W}}{\partial y_i} = (0, 1 - \alpha - \beta)^T$$

The result of four sample Jacobian with respect to four different vertices are shown in the Figure 7. As it is shown, the result is a shape-free image with non-zero values in all triangles that that share the corresponding vertex and zero in all the other outer triangles, the Jacobian is zero. Based the above formula, it is obvious that the Jacobian, inside the triangle, varies linearly.

For computing the second Jacobian, we can employ the shape model, replacing the shape value $\mathbf{s}_i$ with the (x_i, y_i) position of ith shape vertex, based on the position of other corresponding vertex positions. If we assume $(x_i, y_i)^T$ as position of the ith vertex, $(x_i^0, y_i^0)^T$ as the position of the corresponding vertex in the mean-shape, $(x_i^{s_j}, y_i^{s_j})^T$ as the position of the corresponding vertex in jth shape model[15]:

$$(x_i, y_i)^T = (x_i^0, y_i^0)^T + \sum_{j=1}^{n} p_j (x_i^{s_j}, y_i^{s_j})^T$$

Thus, the resulting Jacobians are:

$$\frac{\partial x_i}{\partial \mathbf{p}} = \left(x_i^{\mathbf{s}_1}, x_i^{\mathbf{s}_2}, x_i^{\mathbf{s}_3}, ..., x_i^{\mathbf{s}_n}\right) \quad \frac{\partial y_i}{\partial \mathbf{p}} = \left(y_i^{\mathbf{s}_1}, y_i^{\mathbf{s}_2}, y_i^{\mathbf{s}_3}, ..., y_i^{\mathbf{s}_n}\right)$$

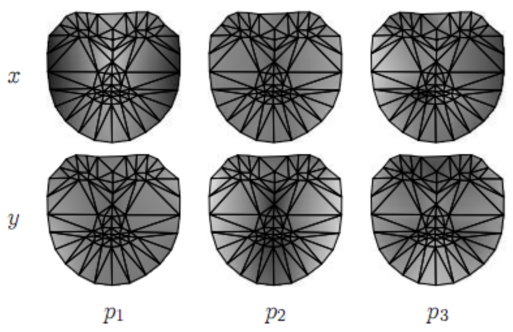
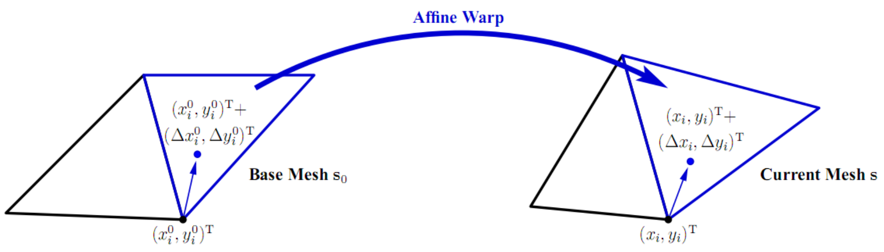


Figure 8: Warp Jacobians $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ [4].



Affine Warp
$(x_i^0, y_i^0)^{\mathrm{T}}+$
$(\Delta x_i^0, \Delta y_i^0)^{\mathrm{T}}$
Base Mesh $\mathbf{s}_0$
$(x_i^0, y_i^0)^{\mathrm{T}}$
$(x_i, y_i)^{\mathrm{T}}+$
$(\Delta x_i, \Delta y_i)^{\mathrm{T}}$
Current Mesh s
$(x_i, y_i)^{\mathrm{T}}$

Figure 9: Warping the pixels from the base mesh $\mathbf{s}_0$ into the current mesh $\mathbf{s}$.

By combining the two derived values for the Jacobians, we can find the overall Jacobian value, $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$, as the ones shown in the Figure 8.

We define the $\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ as the *Steepest Descent Image*, or **SD**. In fact, **SD**s are result of multiplying $\nabla I = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$ in $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$, calculated in the previous section.

By exploiting the equation 2, we need to have a pragmatic look into applying the warp updating by $\mathbf{W}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{p})^{-1}$. The goal is to apply the variation of the shape parameters $\Delta\mathbf{p}$ in a way that makes our model picture, more similar to the input image. The goal is to change the position of the vertices in the current mesh in a way that its shape become more similar the base shape's. As we know by the shape parameter variations $\Delta\mathbf{p}$ we can compute the variation of shape model in the base mesh by employing the basic shape model:

$$\mathbf{s}_0 + \Delta\mathbf{s}_0 = \mathbf{s}_0 + \sum_{i=1}^{n} \Delta\mathbf{p}_i \mathbf{s}_i$$

But in order to apply the shape variations in the current mesh, as shown in the Figure 9, we should calculate the shape variations in the current shape $\mathbf{s}_0$. The problem arousing here is that because of various warp functions defined for each triangle, each of which result in a different value for shape variations in the current mesh. For the sake of overwhelming this challenge we simply calculate various values of the shape variations and then calculate their average to derive the $\Delta\mathbf{s}$. After calculation of $\Delta\mathbf{s}$, we should calculate the new shape parameters $p_i^{'}$:

$$p_i^{'} = \mathbf{s}_i.(\mathbf{s} + \Delta\mathbf{s} + \mathbf{s}_0)$$

where the shape principle components are orthonormal; the . is dot product, and the updated shape parameters are $\{p_i^{'}\}_{i=1}^{n}$.

From now on, we focus on the question that how it is possible to apply the *Inverse Compositional* method for using within AAM algorithm. As it is mentioned before, not only the model is dependent on the shape parameters, $\{p_i\}_{i=1}^{n}$, but also it is dependent upon the appearance parameters $\{\lambda\}_{i=1}^{m}$. After completing the fitting algorithm for shape model, it is the time to derive the generalization of *Inverse Compositional* method for AAM fitting with regard to to the both shape and appearance parameters. To make it clear, we should minimize the following error function with respect to both shape and appearance parameters:

$$E_{\mathbf{p},\lambda} = \sum_{\mathbf{x} \in \mathbf{S}_0} \left[A_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]^2 \tag{3}$$

There are lots of methods introduced for minimization of the above error function, explanation of which are brought in [4] and [13]. Two of the most eminent algorithms are called Project Out(PO) and Simultaneous Inverse Compositional(SIC) which are explained in [4]. The PO algorithm is fast enough for real-time implementations, but the SIC algorithm is more reliable, but slower.

2.2.5 Cutting-edge trends in AAM

The AAM is one of the hot current topics in todays image processing trends. There has been a bunch of works in AAM, that try to reach to an optimal 3D model of the AAM [16, 17, 18, 19, 20]. The other challenge in the kingdom of the AAM is finding some efficient methods for tracking in presence of occlusion[21]. The sister area is illumination-independent AAM which is considered in [22, 23]. The other cutting-edge research on AAM models is the answer of the question "how is it possible to make the algorithm generalize the tracking model over a broad number of faces(deformable objects)", i.e. increasing the generalization power of the algorithm [24, 25, 26, 27, 28]. The papers also consider other challenges like computational complexity, finding the global optimum point of the model. During online time, the significant time burden is due to the complexity of the fitting algorithm. Thus, the faster fitting algorithm one can find, the faster it is possible to process a video. This problem is mostly considered in [29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44]. Yet lots of efforts dedicated to this area, there is still a need for better algorithms both in terms of number of iterations needed for convergence and the accuracy of the fitting. The other disparate trend is confronting the is online-time updating of a defective model based on image construction error; due to general deformation of the objects in long-term, i.e. aging, such methods can be so appealing. The other application of such methods can be about deployment of the algorithm with various ilk of faces, i.e. the system accepts an initial model; after replacing the main operator, the algorithm becomes more adapted to the new dominant faces while working[45, 46].

3 Facial features extraction

Automatic facial feature extraction is one of the most important and attempted problems in computer vision and many multimedia applications. It is a necessary step in face recognition, facial image compression and low-bit video coding. These applications play an important role in security systems, human-computer interaction, and teleconferencing. Facial feature extraction, in general, refers to the detection of eyes, mouth, nose and other important facial components. Various techniques have been proposed in the literature for this purpose, e.g. facial feature points selection[3] and Gabor wavelet based feature extraction utilized in present work.

3.1 Gabor wavelet based feature extraction

The human visual system can be viewed as composed of a filter bank. For the first time the 2D Gabor wavelets were introduced by Daugman [47] for human iris recognition, and Lades et al. [48] exploited Gabor wavelets for face recognition using the Dynamic Link Architecture (DLA) framework. The responses of the respective filters can be modeled by Gabor functions of different frequencies and orientations. The Gabor features have been found to be particularly appropriate for texture representation and discrimination, and have been successfully applied to texture segmentation, face recognition, handwritten numerals recognition, and fingerprint recognition. For face recognition applications, the number of Gabor filters used to convolve face images varies with applications, but usually 40 filters (5 scales and 8 orientations) are used. However, due to the large number of convolution operations of Gabor filters with the image (convolution at each position of the image), the computation cost is prohibitive. Even if a parallel system was used, it took about 7 seconds to convolve a 128*128 image with 40 Gabor filters[48] . For global methods (convolution with the

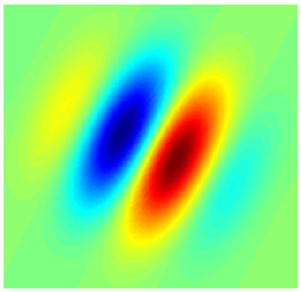


Figure 10: A 2D Gabor filter in spatial domain

whole image), the dimension of the feature vectors extracted is also incredibly large, for example, 163,840 for an image of size 64*64. In the spatial domain, a 2D Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave as follows(Grigorescu et al. 2003; Petkov & Wieling 2006) [49]:

$$G(x,y;\lambda,\theta,\phi,\sigma,\gamma) = e^{\left(-\frac{x_\theta^2+\gamma^2 y_\theta^2}{\sigma^2}\right)} e^{i\left(2\pi\frac{x_\theta}{\lambda}+\phi\right)} \tag{4}$$

where

$$x_\theta = xcos\theta + ysin\theta \tag{5}$$

and

$$y_\theta = -xsin\theta + ycos\theta \tag{6}$$

The parameters used in the above equations are clarified as follows:
1 . σ is the standard deviation of the Gaussian factor and determines the (linear) size of its receptive field.
2 . λ specifies the wavelength of the cosine factor of the Gabor filter.
3 . θ specifies the orientation of the normal to the parallel stripes of the Gabor filter.
4 . ϕ is the phase offset of the cosine factor and determines the symmetry of the Gabor filter.
5 . γ is called the spatial aspect ratio and specifies the ellipticity of the Gaussian factor.
We also use another parameter called the band width (bw) of a gabor filter and is related to the ratio $\frac{\sigma}{\lambda}$ as follows:

$$bw = \log_2 \frac{\frac{\sigma}{\lambda}\pi + \sqrt{\frac{\ln 2}{2}}}{\frac{\sigma}{\lambda}\pi - \sqrt{\frac{\ln 2}{2}}} \tag{7}$$

Figures 10 & 11 show some samples of 2D Gabor wavelet in both spatial and frequency domains.

As mentioned before, in this method we first form a Gabor wavelet bank in different frequencies, scaling and orientations and apply those Gabor wavelets to an image and create coefficients that can then be used to reconstruct the image and in the rest of report we will state that these coefficients also can be used as our extracted features. The Matlab code involves two main parts. The first part loops through all the pixels of the image and multiplies the Gabor wavelet to each of them, and the second part alter orientation and bandwidth of the wavelet to make different scaling and orientation. Moreover, it is noticeable that all the other parameters are constant throughout the process.
Figures 12 & 13 & 14 represent the Gabor wavelet bank , the original image and output images very well. As explained, according to the lots of former works we considered Gabor wavelet bank in 8 orientations and 5 scaling.

It is noticeable that the magnitude of output images were depicted in figure 14 and the value of magnitudes can be used as our extracted features. The process of extracting features are what will clarify as follows. As mentioned, once Gabor wavelets have been designed, image features at different location, frequency and orientation can be extracted by convolving the image $I(x,y)$ with the filters and calculating the magnitude of the result as below:

$$O(x,y;\lambda,\theta,\phi,\sigma,\gamma) = |I(x,y) * G(x,y;\lambda,\theta,\phi,\sigma,\gamma)| \tag{8}$$

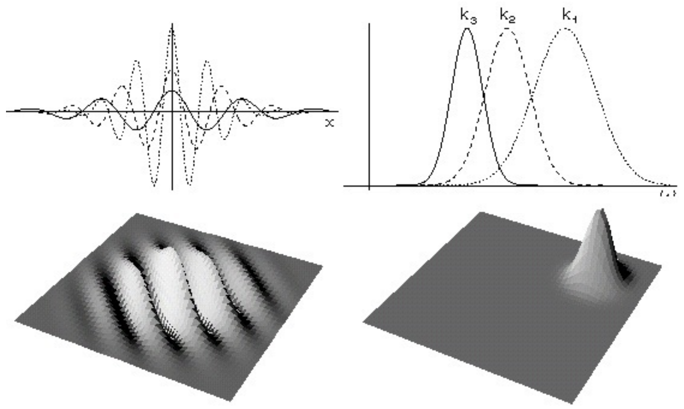


Figure 11: Spatial domain[left image], frequency domain[right image]

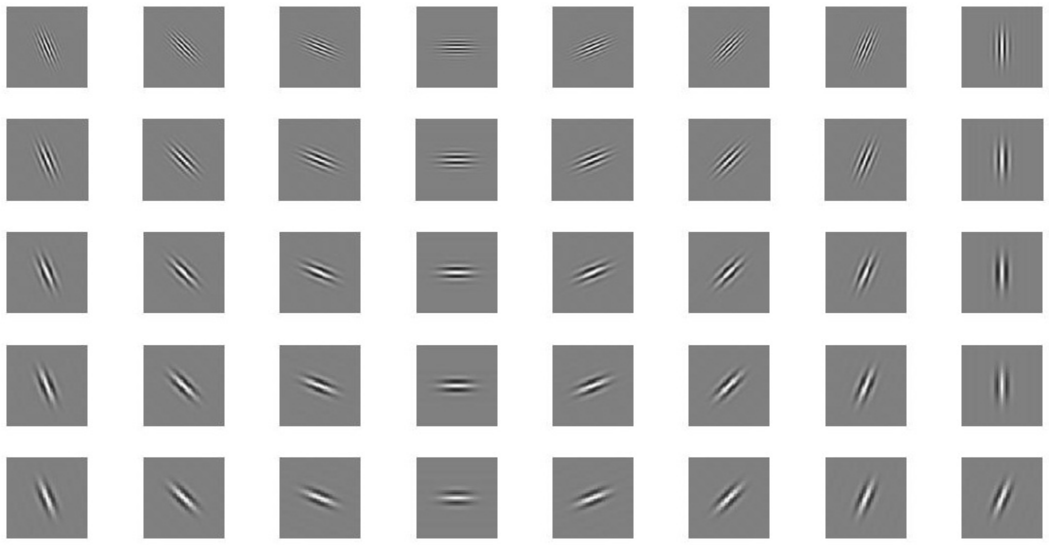


Figure 12: Gabor wavelet bank in 5 frequencies and 8 orientations

Figure 13: Original image in 50*50

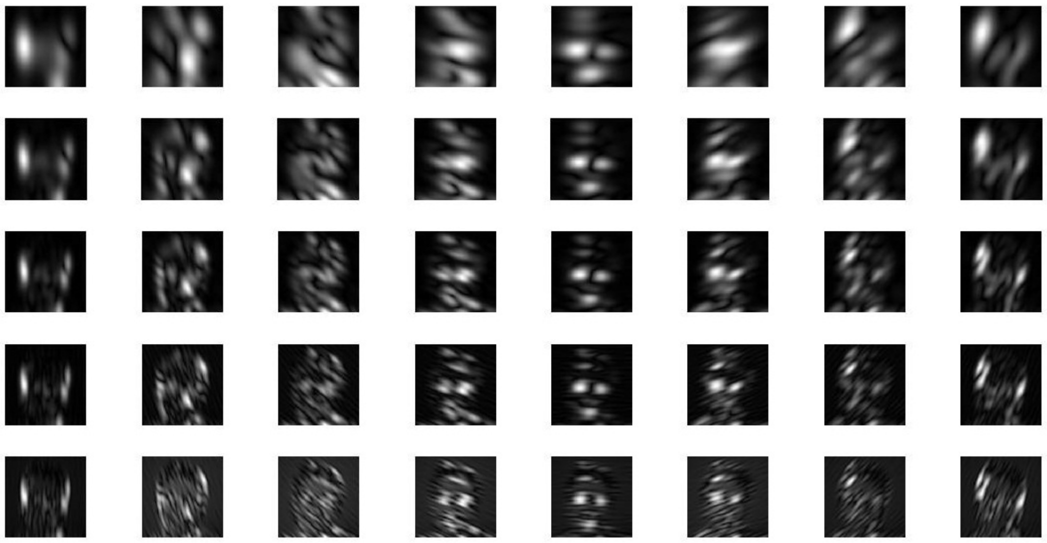


Figure 14: Magnitude of output images of Gabor wavelet

Figure 14 shows the magnitudes of Gabor representation of a face image with 5 scales and 8 orientations. A series of row vectors could be obtained out of $O(x, y)$ by concatenating its rows or columns, which are then concatenated to generate a discriminative Gabor feature vector:

$$\text{Feature vector}= (O^1(x, y), O^2(x, y), ..., O^{40}(x, y))^T \tag{9}$$

Where $O^i(x, y)$ is the ith concatenated matrix over all its rows.

3.2 Feature points selection based feature extraction

After tracking of feature points from the first frame up to the last one, according to the [3] thirteen features are extracted from the normalized position of the feature points by the position of the tip of the nose in the first frame and the last one. Afterwards, these calculated features build a feature vector for a set of sequences of gestures which form a command as explained in introduction. Then the extracted feature vector is applied to multi-class multi-kernel relevance vector machine. According to figure 15 the extracted features are illustrated in Table 1.

As mentioned before, the calculated value of the above 13 extracted features can make a feature vector with 13 elements for all sequences of gestures corresponded to each command. Hence, due to the low dimension of feature vector in comparison with the last method (Gabor wavelet based) we don't need to use feature selection techniques so as to reduce the dimension of feature vector, therefore the lower complexity of system will be achieved.

4 Feature selection

Due to the high dimension of feature vector generated by Gabor wavelets, to select the relevant global and local appearance features with the most discriminating information, we need to use some techniques in order to selectivity reduce the dimensionality of the feature space that in turn results in significant speed up during on-line classification. we should recognize from a high dimensional feature space only those dimensions that convey the most information making the feature extraction process during the testing phase very efficient with lower complexity. There are some methods in literature which proposes novel techniques to do that.

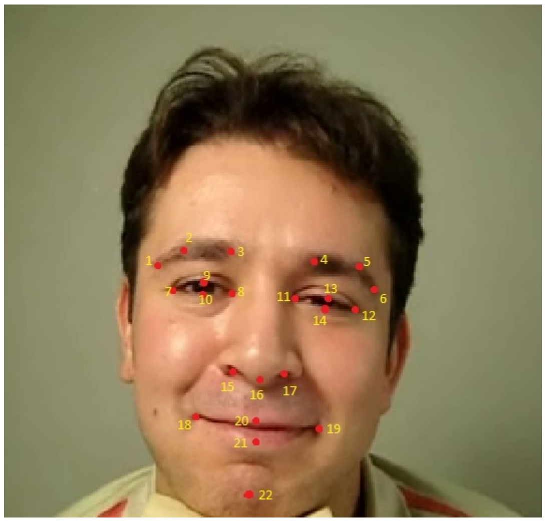


Figure 15: The 22 facial feature points

Genetic Algorithm (GA) using selection of Gabor filters for pixel classification [50] and other classification applications. However, the computation cost of GAs is very high, particularly in the case when a huge number of features are available like our work. In addition, recently the AdaBoost algorithm has been used to learn the most discriminative Gabor features for classification [51]. Unlike its success, AdaBoost algorithm selects only features which perform individually best, the redundancy among all generated features is not considered. So it may choose features with the same information without considering them with each other.[52]. Information theory techniques are another approach so as to feature selection. For instance, Conditional mutual information based method for selecting Gabor features for face recognition[53, 54] have been used. Furthurmore, another novel method that recently has been used is based on Imperialist Competitive Algorithm (ICA) done by M.H. Sigary and Caro Lucas, 2009. Moreover, there are some methods in literature based on Principle Component Analysis (PCA). Applying Principal Component Analysis (PCA) can reduce the dimensionality without sacrificing performance [55]. But, one can not avoid a high dimensional feature space altogether since the principal components in PCA are still linear combinations of the original features. Present report is aiming to utilize 2 common methods for feature selection, first, multi class AdaBoost, second, principle feature analysis (PFA), based on principle component analysis (PCA). It didn't matter to us, using gabor wavelet so as to featre extraction and sacrificing the computation cost, on account of the fact that it has been proven that the Gabor wavelet representation of face images is robust against variations due to illumination and facial expression changes.

4.1 Conventional AdaBoost classifier

AdaBoost is an profitable classification algorithm, which expressively has been used for classification and feature selection applications in literature. It works based on making a linear combination of weak classifiers (features) and building a strong classifier (feature). The training data contains clients examples and impostors examples with the AdaBoost parameter $y = 0, 1$ for impostors and clients, respectively. The initial weight $w_{1,i}$ for each example is given according to the number of client's or impostors examples. All weights of clients examples are set equally, so as to impostors examples. The number of clients is L, and the number of impostors is M. The initial weight $w_{1,i}$ for clients is $1/2l$, and for impostors is $1/2m$. Each Gabor wavelet feature j corresponds to a weak classifier h_j. A Linear Fisher Discriminant (LFD) classifier is adopted as the weak classifier in this review. The LFD classifier determines the optimal threshold classification function, such that the minimum number of examples is misclassified. When all the classifiers are trained, the weak classifier h_t is selected with the lowest classification error ε_t. After each round of training, the weights $w_{t,i}$ of the training data are modified in order to emphasise those examples which are misclassified by the previous

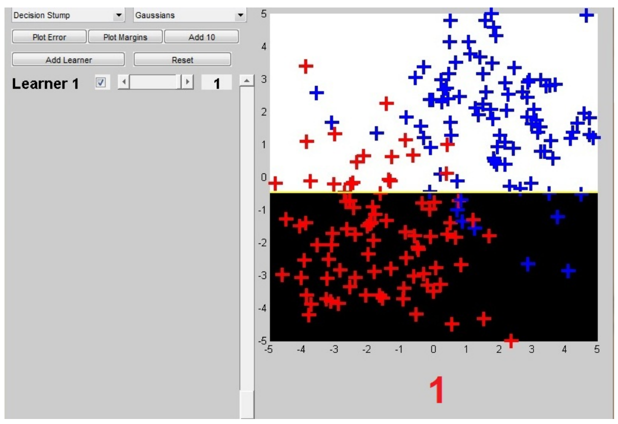
Table 1: Extracted features based on the position of feature points

Number	Feature	Calculated value
1	Width of Eyes	$we = \frac{(x_8-x_7)+(x_{12}-x_{11})}{2}$
2	Openness of Eyes	$oe = \frac{(y_9-y_{10})+(y_{13}-y_{14})}{2}$
3	Height of Eyebrows 1	$he1 = \frac{(y_3-y_{16})+(y_4-y_{16})}{2}$
4	Height of Eyebrows 2	$he2 = \frac{(y_1-y_{16})+(y_6-y_{16})}{2}$
5	Width of Mouth	$wm = x_{19} - x_{18}$
6	Openness of Mouth	$om = y_{20} - y_{21}$
7	Nose tip and Lip corners distance	$nl = \frac{(y_{16}-y_{18})+(y_{16}-y_{19})}{2}$
8	Chin and Lip corners distance	$cl = \frac{(y_{18}-y_{22})+(y_{19}-y_{22})}{2}$
9	Nose corners and Chin distance	$nc = \frac{(y_{15}-y_{22})+(y_{17}-y_{22})}{2}$
10	Nose corners and Eyebrows distance	$ne = \frac{(y_3-y_{15})+(y_4-y_{17})}{2}$
11	Openness to Width ratio of mouth	$ow = \frac{OM}{WM}$
12	Corners and Bottom points of mouth distance	$cb = \frac{(y_{18}-y_{21})+(y_{19}-y_{21})}{2}$
13	Two Eyebrows distance	$te = x_4 - x_3$

weak classifier. The final strong classifier H takes the form of a weighted combination of weak classifiers h_t followed by a threshold. After feature selection, an individual can be represented by a face model consisting of the selected key features.
We have implemented conventional AdaBoost for classification algorithm is depicted in figure 16 & 17 step-by-step. The error and margin figures are plotted as well. This is a demo of how AdaBoost works. The demo displays some 2-D data, and shows the development of the decision boundary, margins, error rate and margin distribution as a boosted ensemble is trained. The data is split into two classes, red and blue. The decision boundary appears as a yellow contour. The margins are shown on a white-black gradient, where white is a confident classification as blue and black is a confident classification as red. In each step, one weak learner is added, and learners are shown in a scrollable list, with their adjusted weights (in linear combination) beside them. Weight adjustment involves normalisation and then scaling so that 1 represents an average learner weight. "Plot Error" shows the training error (blue) and the generalization error (red) against the size of the ensemble. "Plot Margins" shows the CDF of the margins for the current ensemble. The height of the line above 0 indicates the proportion of the data with that margin.

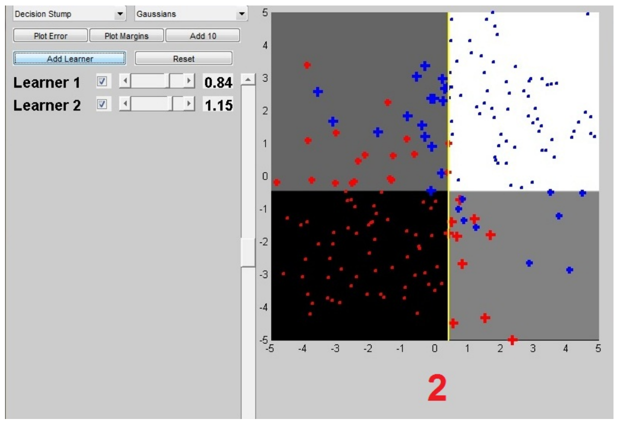
4.1.1 Multi-Class AdaBoost feature selection[1]

AdaBoost (an abbreviation of Adaptive Boosting) was formulated by Freund and Schapire [56]. It is a relatively efficient, simple, and easy learning strategy for improving the performance of classification algorithms. AdaBoost is robust to noisy data, but susceptible to over fitting. It was first applied to face detection by Viola and Jones [57]. In their work, features were extracted by Haar wavelets, then those features most significant for face detection were selected by a modified AdaBoost algorithm. The images were finally classified by a cascade of AdaBoost classifiers. The final system detected faces very quickly. In this report, we use a Gabor wavelet feature based AdaBoost approach for face feature selection. The initial idea is to from a two-class classification concept similar to the work done by Viola and Jones in face detection [57].
For a given image $I(x, y)$ with $N * M$ pixels, the number of Gabor wavelet feature representations will be of the order $N * M * 40$. The feature space consists of all these features. In this case, it is 40 times larger than the original image space. The Gabor wavelet feature representation described above resides in a very high dimensional space. It is important to reduce the feature space to a lower dimensional representation by feature selection. In this report, we use AdaBoost to select significant features from the pool of Gabor wavelet features, forming a strong linear combination of selected features, to reduce the dimension.
AdaBoost is an efficient method for producing a highly accurate learning algorithm by combining a set of rough and moderately accurate learning algorithm. Inaccurate learning algorithms sometimes are called " weak ",learners, " weak " classifiers, or base classifiers. They have lower discrimination power to assign the true label correctly. Hence, AdaBoost refers to a method of combining a set of weak learners into a strong



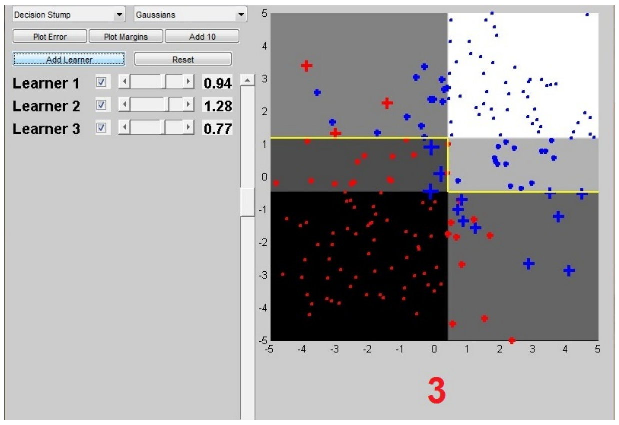
Learner 1 1

1



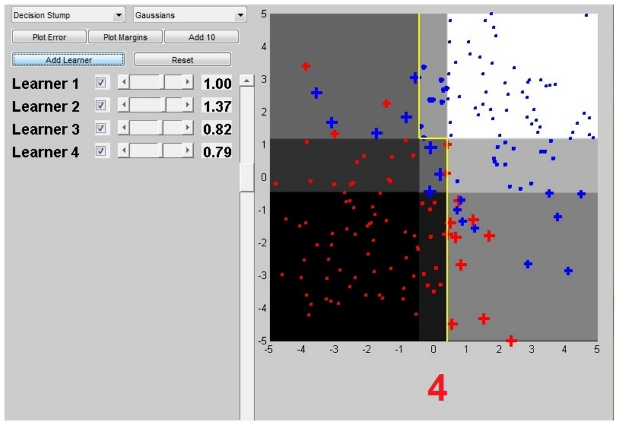
Learner 1 0.84
Learner 2 1.15

2



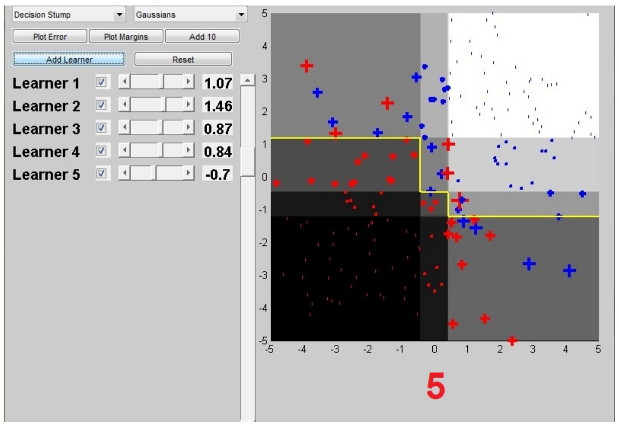
Learner 1 0.94
Learner 2 1.28
Learner 3 0.77

3



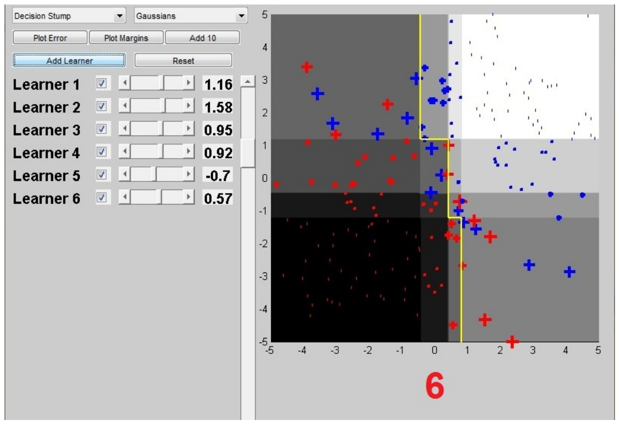
Learner 1 1.00
Learner 2 1.37
Learner 3 0.82
Learner 4 0.79

4



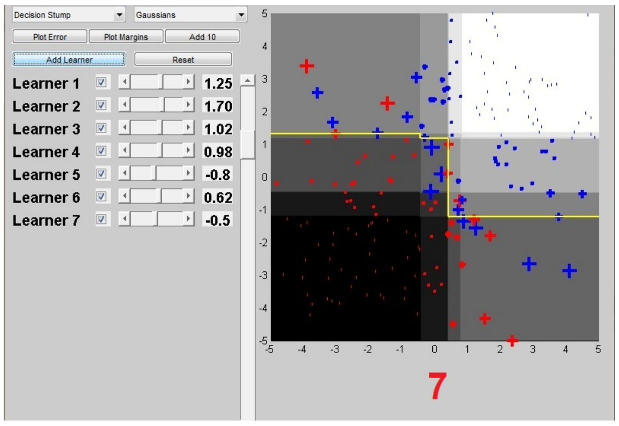
Learner 1 1.07
Learner 2 1.46
Learner 3 0.87
Learner 4 0.84
Learner 5 -0.7

5



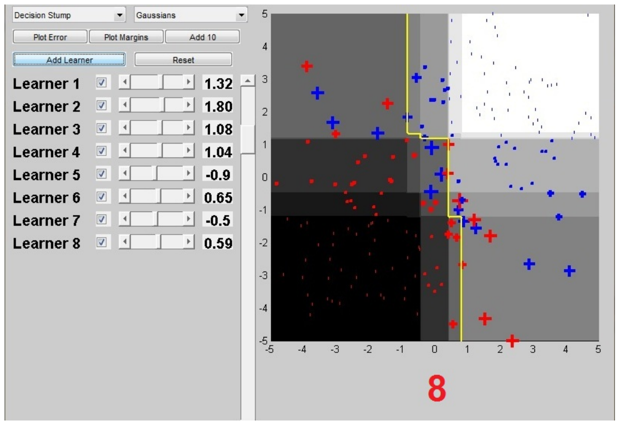
Learner 1 1.16
Learner 2 1.58
Learner 3 0.95
Learner 4 0.92
Learner 5 -0.7
Learner 6 0.57

6



Learner 1 1.25
Learner 2 1.70
Learner 3 1.02
Learner 4 0.98
Learner 5 -0.8
Learner 6 0.62
Learner 7 -0.5

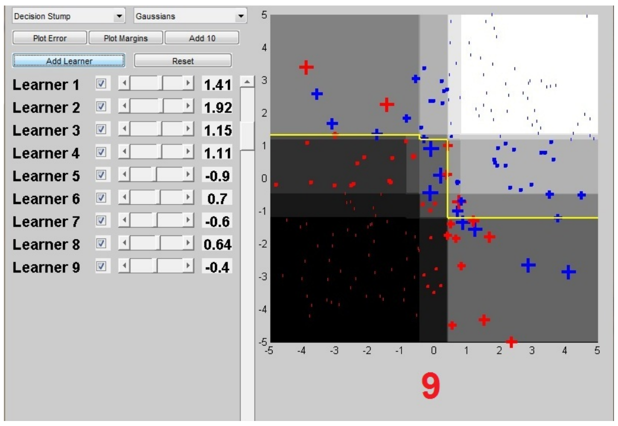
7



Learner 1 1.32
Learner 2 1.80
Learner 3 1.08
Learner 4 1.04
Learner 5 -0.9
Learner 6 0.65
Learner 7 -0.5
Learner 8 0.59

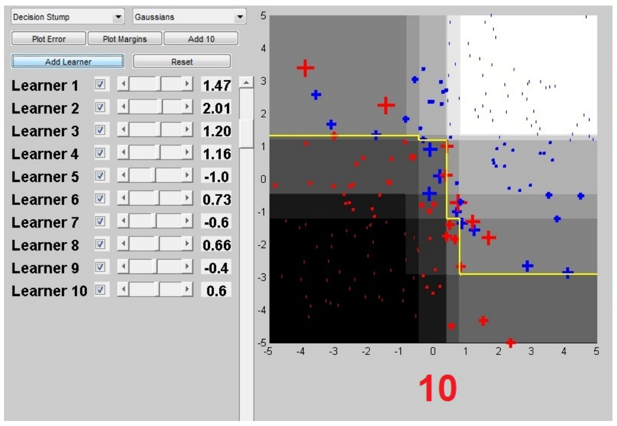
8

Figure 16: The process of AdaBoost classification of 2 classes of generated Gaussian data (a)



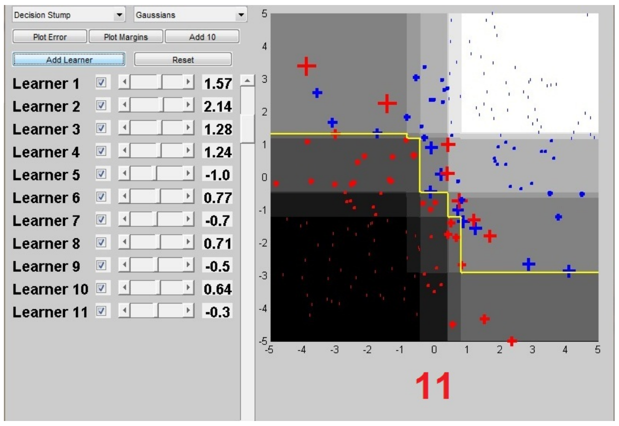
```
Decision Stump    Gaussians
Learner 1   1.41
Learner 2   1.92
Learner 3   1.15
Learner 4   1.11
Learner 5   -0.9
Learner 6   0.7
Learner 7   -0.6
Learner 8   0.64
Learner 9   -0.4
```

9



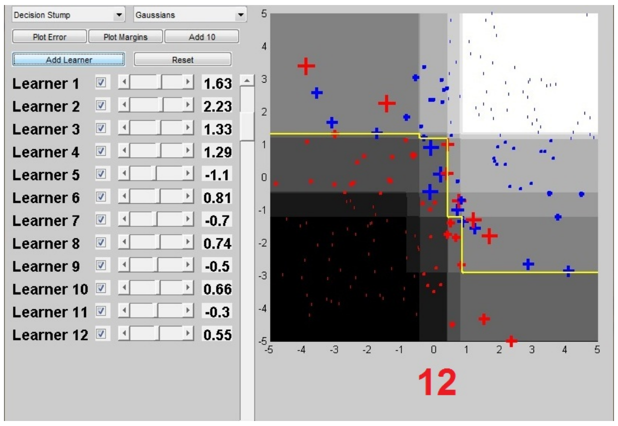
```
Decision Stump    Gaussians
Learner 1   1.47
Learner 2   2.01
Learner 3   1.20
Learner 4   1.16
Learner 5   -1.0
Learner 6   0.73
Learner 7   -0.6
Learner 8   0.66
Learner 9   -0.4
Learner 10  0.6
```

10



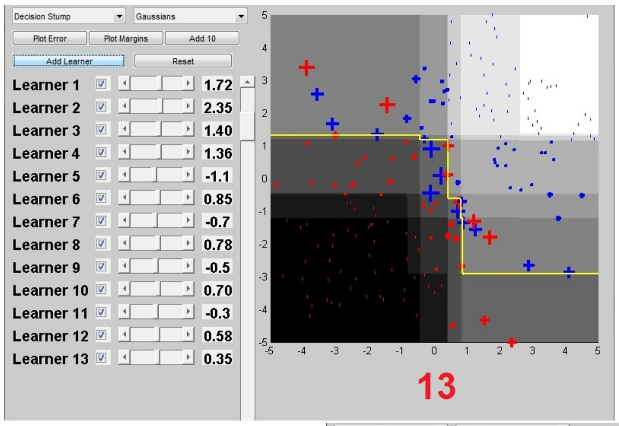
```
Decision Stump    Gaussians
Learner 1   1.57
Learner 2   2.14
Learner 3   1.28
Learner 4   1.24
Learner 5   -1.0
Learner 6   0.77
Learner 7   -0.7
Learner 8   0.71
Learner 9   -0.5
Learner 10  0.64
Learner 11  -0.3
```

11



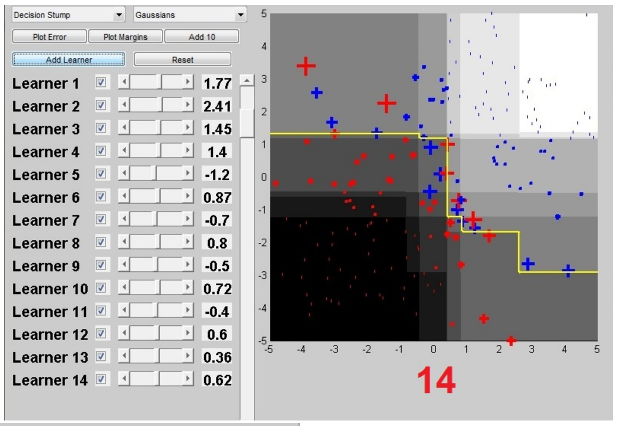
```
Decision Stump    Gaussians
Learner 1   1.63
Learner 2   2.23
Learner 3   1.33
Learner 4   1.29
Learner 5   -1.1
Learner 6   0.81
Learner 7   -0.7
Learner 8   0.74
Learner 9   -0.5
Learner 10  0.66
Learner 11  -0.3
Learner 12  0.55
```

12



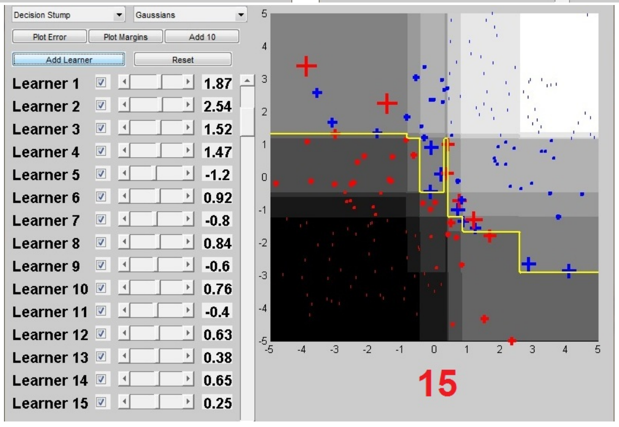
```
Decision Stump    Gaussians
Learner 1   1.72
Learner 2   2.35
Learner 3   1.40
Learner 4   1.36
Learner 5   -1.1
Learner 6   0.85
Learner 7   -0.7
Learner 8   0.78
Learner 9   -0.5
Learner 10  0.70
Learner 11  -0.3
Learner 12  0.58
Learner 13  0.35
```

13



```
Decision Stump    Gaussians
Learner 1   1.77
Learner 2   2.41
Learner 3   1.45
Learner 4   1.4
Learner 5   -1.2
Learner 6   0.87
Learner 7   -0.7
Learner 8   0.8
Learner 9   -0.5
Learner 10  0.72
Learner 11  -0.4
Learner 12  0.6
Learner 13  0.36
Learner 14  0.62
```

14



```
Decision Stump    Gaussians
Learner 1   1.87
Learner 2   2.54
Learner 3   1.52
Learner 4   1.47
Learner 5   -1.2
Learner 6   0.92
Learner 7   -0.8
Learner 8   0.84
Learner 9   -0.6
Learner 10  0.76
Learner 11  -0.4
Learner 12  0.63
Learner 13  0.38
Learner 14  0.65
Learner 15  0.25
```

15

Figure 17: The process of AdaBoost classification of 2 classes of generated Gaussian data (b)

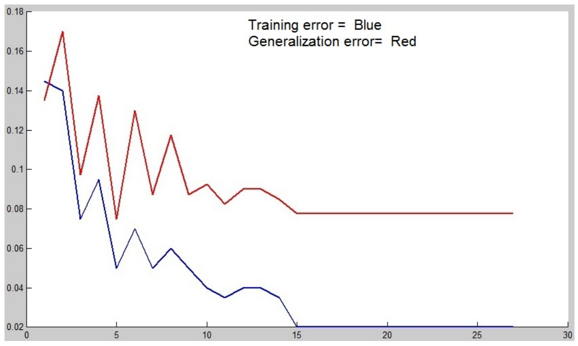


Figure 18: The training error (blue) and the generalisation error (red) against the size of the ensemble

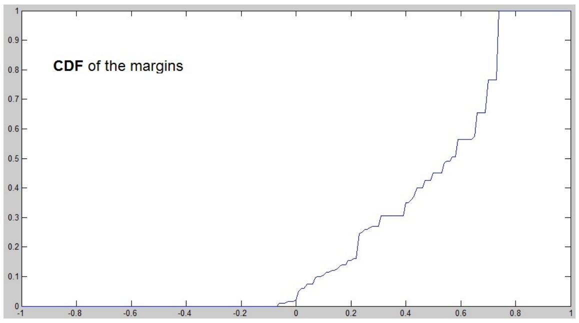


Figure 19: CDF of the margins for the current ensemble

classifier which gives a high accuracy for prediction and classification. AdaBoost has been a very prosperous approach for solving two-class classification. It also has two multi-class classification versions, AdaBoost.$M1$ [56] and AdaBoost.$M2$ [56]. $M1$ is the most direct way to perform multi-class classification, and $M2$ is an enhancement of $M1$. The multi-class AdaBoost algorithm is a combination of $M1$ and $M2$ which is a variant from $M1$ with some extension on multiple labels from $M2$. The algorithm of multi-class AdaBoost is given in Table 6. In each iteration, a significant feature is selected with the lowest error ε_t so that there are T significant features after T iterations. A multi-class weak learner $mh(x)$ is built on a single feature. The error ε is calculated by summing the weights of all misclassified examples. The importance α_t for each significant feature is evaluated by the lowest error ε_t in the t-th iteration.

4.1.2 Multi-Class weak learner

In AdaBoost learning, weak learner is used to evaluate a feature. In [58], it has demonstrated that different weak learners lead to different performance of AdaBoost at the end. Hence, weak learners are crucial and primitive parts of the algorithm, and the design of weak learner is very important. In this report, a multi-class weak learner - mPotsu (multi-class Potsu) is skimmed. The mPotsu weak learner is a multi-class variant from Potsu weak learner, which is a type of two-class classifier built with the concept of perceptron [59] and Otsus thresholding algorithm [60]. The mPotsu is constructed by multiple Potsu weak learners with the one-against- rest strategy [61]. The strategy adopts a set of binary classifiers to create a multi-class classifier. Given an example x and its label in k classes, a binary Potsu weak learner is trained between a class j and test $k-1$ classes. Since there are k classes, an mPotsu contains k binary Potsu weak learners. Each Potsu gives an output indicating examples belonging to class j or not. Each Potsu in a mPotsu is built for a particular class (client) against rest classes (clients). Given an example x, the first Potsu tells x whether the first client or not. The input vector of mPotsu is taken from a single Gabor wavelet feature across all examples in the training set. Hence, the input vector is one-dimensional. The training in each Potsu uses a heuristic approach to find an optimal threshold for separating the examples from the class j

Algorithm 6 Multi-Class AdaBoost algorithm

```
1:  Given example $(x_1, y_1), ..., (x_n, y_n)$, where $x_i$ is the data of the $i$th example,
    which are contributed by $k$ features $\{j_1, ..., j_k\}$,
    and $y_i \in Y = \{1, ..., c\}$ for $c$ subjects (classes).
2:  Initialize the weights $w_{1,i} = \frac{1}{n}$ for each example $(x_i, y_i)$.
3:  for $t = 1, ..., T$ do
4:      Normalize the weights $w_{t,i} \Longleftarrow \frac{w_{t,i}}{\sum_{i=1}^{n} w_{t,i}}$ so that $w_t$ form a probability distribution.
5:      for all $\{j_1, ..., j_k\}$ do
6:          Train a multi-class weak learner $mh_j$ built with one single
            feature $j$ with the weights $w_{t,i}$.
7:          the error is calculated as $\varepsilon_j = \sum_{i=1}^{n} w_{t,i}\gamma$, where $\gamma = 1$
            when $y_j \notin mh_t(x_i)$, and $\gamma = 1$ otherwise.
8:      end for
9:      Choose the optimal multi-class weak learner $mh_t$ with the
        lowest error $\varepsilon_t$ from all $mh_j$.
10:     Select the corresponding feature $j_t$ of the multi-class weak
        learner $mh_t$ as a significant feature.
11:     Remove the feature $j_t$ from the feature set $\{j_1, ..., j_k\}$.
12:     Update the weights $w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$, where $e_i = 0$ if
        $y_j \notin mh_t(x_i)$ and $e_i = 1$ otherwise, and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.
13: end for
```

and other examples. If an example is successfully classified, the output will be labelled as 1, i.e. the positive. If not, the output will be labelled as 0, i.e. the negative. Ideally, if the training examples are well separated between each class, there will be only one positive output and the rest of samples will be negative outputs. The plausible label is the corresponding class of positive label in mPotsu. However, in practice, there may be more than one Potsu weak learners giving positive labels. It is because the training examples are not linearly separable between different classes in most real situations. Hence, in the case of one-against-rest strategy, it is rare to output only one plausible label of multi-class classifiers. Instead of giving only one absolute label, mPotsu outputs a label vector η which compromises multiple positive labels.

$$\eta = (l_1, l_2, ..., l_k) \tag{10}$$

The label vector η contains k components corresponding to k classes. Each component l_j is corresponding to each class j, and has a boolean value between 1 and 0 indicating acceptance or rejection on the corresponding class j. By introducing the label vector η, mPotsu is able to coexist multiple decisions. For example, an mPotsu gives a label vector $\eta = (1, 1, 1, 0, 0, ..., 0)$, which indicates the plausible class might be class 1, class 2 or class 3 (since $l_1 = 1$, $l_2 = 1$, and $l_3 = 1$). It makes mPotsu not give an absolute decision, but several possible decisions. In AdaBoost, after every weak learner is trained, an evaluation is needed to test the performance of the trained weak learner. Due to multi-label complexity on multi-class classification, a loose evaluation method is applied in mPotsu. Given an example with its true class j, if $l_j = 1$ in the giving label vector η, the classification will be considered as true positive no matter what other components $l_i = 1$. The classification is defined as:

$$y \in mh_t(x) \; when \; l_y = 1 \tag{11}$$

where $mh_t(x)$ represents the mPotsu weak learner. For instance, an example x with its true label of class 1 is fed into an mPotsu. After training, the mPotsu gives a label vector $\eta = (1, 1, 1, 0, ..., 0)$, where l_1 in the vector η is equal to 1, the classification is considered as true regardless $l_1 = 1$, $l_2 = 2$, and $l_3 = 3$. Since the method of classification is quite loose, the accuracy of mPotsu is low through assigning multiple labels to an example. However, since AdaBoost can boost the performance from a set of weak classifiers, the low accuracy on different mPotsu is accumulated into higher accuracy. After many iterations, the performance of multi-class AdaBoost is enhanced.

4.1.3 Substantial computation cost

Feature selection on multi-class AdaBoost is very time consuming. Having N image sets, N Potsu binary learners we would have for each mPotsu. The computational time on training an mPotsu is equivalent to time cost on training N binary Potsu learners individually. The AdaBoost algorithm searches over the whole feature set exhaustively in each iteration. For instance, if the size of each face image is $27 * 28$, and these images are convolved with 40 Gabor wavelet kernels in feature extraction to make magnitude responses, the total number of Gabor wavelet features is $27 * 28 * 40 = 30240$, so by having 800 face images in dataset, With a 2.8 GHz CPU, the training on a single mPotsu needs 11 seconds. One iteration in AdaBoost training takes $11 * 30240 = 332640$ seconds which is roughly 93 hours. To select 200 features, it needs 200 iterations. The AdaBoost training takes $93 * 200 = 18600$ hours, i.e. 775 days [1]. The whole computational time is extremely long, and it makes feature selection hardly being accomplished with current computing facility.

4.2 Principal feature analysis based on PCA (principal component analysis)

In pattern recognition and general classification problems, methods such as principal component analysis (PCA), independent component analysis (ICA) and Fisher linear discriminate analysis (LDA) have been extensively used. These methods find a mapping between the original feature space to a lower dimensional feature space.
In some applications it might be desired to pick a subset of the original features rather then find a mapping that uses all of the original features. The benefits of finding this subset of features could be in cost of computations of unnecessary features, cost of sensors (in physical measurement systems).
The optimality properties of PCA have attracted research on PCA based variable selection methods [62, 63, 64, 65]. But these methods have the disadvantage of either being too computationally expensive, or choosing a subset of features with redundant information. This report investigates a computationally efficient method that exploits the structure of the principal components of a feature set to find a subset of the original feature vector. The chosen subset of features is shown empirically to maintain some of the optimal properties of PCA. The following is a brief review of the existing PCA based feature selection methods.
By a linear transform, a random vector $\mathbf{X} \in \Re^{\mathbf{n}}$ with zero mean and covariance matrix $\sum_{\mathbf{x}}$ can be mapped to a lower dimension random vector $\mathbf{Y} \in \Re^{\mathbf{q}}$, $q < n$:

$$Y = A_q^T X \tag{12}$$

$$A_q^T A_q = I_q \tag{13}$$

Where I_q is the $q \times q$ identity matrix.
For the PCA technique, A_q is a $n \times q$ matrix whose columns are the q orthonormal eigenvectors corresponding to the first q dominant eigenvalues of the covariance matrix $\sum_x$. One of the important properties of this technique is the maximization of the "spread" of the points in the lower dimensional space which means that the points in the mapped space are kept as far apart as possible, retaining the variation in the original space. We can see the minimization of the mean square error between the predicted data to the original data as well. Now we are aiming to choose a subset of the original variables/features of the random vector X. The linear transform of X can be viewed as follows:

$$\begin{pmatrix} I_q \\ 0_{(n-q)\times q} \end{pmatrix}$$

Any permutation of the rows of A_q as another matrix is acceptable. Without loss of generality, we can rewrite the corresponding covariance matrix of X as follows:

$$\begin{pmatrix} \{\sum_{11}\}_{q\times q} & \{\sum_{12}\}_{q\times(n-q)} \\ \{\sum_{21}\}_{(n-q)\times q} & \{\sum_{22}\}_{(n-q)\times(n-q)} \end{pmatrix}$$

This method is very appealing since it satisfies well-defined properties. The drawback of this method is in the complexity of finding the subset to check all combinations of subsets of features so as to find one which satisfies the properties. It is not computationally feasible to find this subset for a large feature vector. In order to minimize the mean square prediction error, we are supposed to minimize the below trace:

$$\sum_{22|1} = \sum_{22} - \sum_{21} \sum_{11}^{-1} \sum_{12} \tag{14}$$

And the retained variability of a subset can be measured using the following equation:

$$Retained\ Variability = (1 - \frac{trace(\sum_{22|1})}{\sum_{i=1}^{n} \sigma_i^2}) \tag{15}$$

where σ_i is the standard deviation of the $i - th$ feature.
Another method, proposed in [63], uses the principal components as the basis for the feature selection. A high absolute value of the $i - th$ coefficient of one of the principal components implies that the x_i element of X is very dominant in that axes/PC. By choosing the variables corresponding to the highest coefficients

of each of the first q PCs, the same projection as that computed by PCA is approximated. This method helps reduce the redundancy of informations, also is a very intuitive and computationally feasible method. However, because it considers each PC independently, variables with similar information content might be chosen.
In PFA (principal feature analysis), it exploits the information that can be inferred by the PC coefficients to obtain the optimal subset of features, despite the basis PCA, all of the PCs are used together to gain a better insight on the structure of our original features so variables can be chosen without redundancy of information.

4.2.1 Concepts of principal feature analysis (PFA) [2]

Consider vector X built from the main feature vector to be a zero mean n-dimensional vector, $\sum$ to be the covariance matrix of X and A to be a matrix whose columns are the orthonormal eigenvectors of the matrix $\sum$:

$$\sum = A\Lambda A^T And A^T A = I_n \tag{16}$$

$$\Lambda = \begin{pmatrix} \lambda_1 & & \\ & . & 0 \\ 0 & & . \\ & & \lambda_n \end{pmatrix}$$

where Λ is a diagonal matrix whose diagonal elements are the eigenvalues of $\sum$, $\lambda_1 \geq \lambda_2 \geq, ..., \geq \lambda_n$. Let A_q be the first q columns of A_q and let $V_1, V_2, ..., V_n \in \Re^q$ be the rows of the matrix A_q. Each vector V_i represents the projection of the $i-th$ feature (variable) of the vector X to the lower dimensional space, that is, the q elements of V_i correspond to the weights of the $i-th$ feature on each axis of the subspace. The key observation is that features that are highly correlated or have high mutual information will have similar absolute value weight vectors V_i (it is shown tha changing the sign has no statistical significance [63]). On the two extreme sides, two independent variables have maximally separated weight vectors, whereas two fully correlated variables have identical weight vectors (up to a change of sign). To find the best subset we can use the structure of the rows V_i to find the first subsets of features that are highly correlated and follow to choose one feature from each subset. The chosen features represent each group optimally in terms of high spread in the lower dimension, reconstruction and insensitivity to noise. The algorithm can be summarized in the following five steps illustrated in table 7. For more clarification it should be noted that the clustering is the representation of the features in the lower dimensional space, and not of the projection of the measurements to that space (as in [64]).The complexity of the algorithm is of the order of performing PCA, because the $K-Means$ algorithm is applied on just n $q-$dimensional vectors.

Algorithm 7 Explanation of principal feature analysis step-by-step

```
1: Compute the sample covariance matrix, or use the true covariance matrix if it is
   available. In some cases it is preferred to use the correlation matrix instead of the
   covariance matrix [63]. The correlation matrix is defined as
   the $n \times n$ matrix whose $i, j - th$ entry is as shown in equation 17:
```

$$\rho_{i,j} = \frac{E[X_i X_j]}{E[X_i^2]E[X_j^2]} \tag{17}$$

```
2: Compute the Principal components and eigenvalues of the
   Covariance/Correlation matrix as defined in equation 16.
3: Choose the subspace dimension $q$ and construct the matrix $A_q$ from $A$. This can
   be chosen by deciding how much of the variability of the data is desired to be
   retained. The retained variability is the ratio between the sum of the first $q$
   eigenvalues and the sum of all eigenvalues, so it can be computed as follows:
```

$$Retained\ Variability = \frac{\sum_{i=1}^{q} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \times 100\% \tag{18}$$

```
4: Cluster the vectors $|V_1|, |V_2|, ..., |V_n| \in \Re^q$ to $p \geq q$ clusters using K-Means
   algorithm. The distance measure used for the $K - Means$ algorithm is the
   Euclidean distance. Choosing $p$ greater than $q$ is usually necessary if the same
   variability as the $PCA$ is desired (usually $1 - 5$ additional dimensions are needed).
5: For each cluster, find the corresponding vector $V_i$ which is closest to the mean of
   the cluster. Choose the corresponding feature, $x_i$ , as a principal feature. This
   step will yield the choice of $p$ features. The reason for choosing the vector
   nearest to the mean is twofold. This feature can be thought of as the central
   feature of that cluster, the one most dominant in it, and which holds the least
   redundant information of features in other clusters. Thus it satisfies both of the
   properties we wanted to achieve, large "spread" in the lower dimensional space,
   and good representation of the original data.
```

5 Conclusions

In this report, we summarized a general review of what we have studied in our summer intern-ship. As mentioned before, in order to do analysis on various parts of face, it is mandatory to recognize and track features on face. In order to reach to such goal we first developed the idea of correlation-based tracking, with two extensions, which was not sufficient for our project's demands. Thus, we moved on another algorithm, namely Active Appearance Model which makes a dynamic model of face. Based the available resources, we derived and implemented the algorithm for our own set of images. The preliminary results of AAM implementation were herald of a fast enough facial feature tracking algorithm.
Simultaneously with facial feature tracking, we worked on some methods for extracting facial features, the most important of which was employment of the Gabor wavelet. At the rest of the feature extraction trend, due to curse of dimensionality in the number of the features, we moved on selecting an optimum and discriminative set of features.

As a future work, we will continue working on various refinements in different parts of this project. At the rest of our trend, one of the fundamental aims is employing Multi-Kernel Multi-Class Relevance Vector Machine, for adaptive generation and recognition of facial commands. As explained before the term *facial commands* is different from gestures or expressions on face, instead a combination of both during time. Thus, we will include the time parameter in our model.

6 Acknowledgement

The authors are delighted to acknowledge their supervisors Mr. Mahdi Kalayeh and Dr. Hamid Sheikhzadeh for their inspiring ideas, comments and guidances on this multifarious project. The authors also acknowledge the research-aimed non-commercial source code releases in [66], [67] and [68]; some parts of each of the implementations were used in part, in our own implementation.

References

[1] M. Zhou, H. Wei, I. Bland, A. Worrall, D. Spence, X. Wang, P. Wen, and F. Liu, "Multi-class adaboost learning of facial feature selection through grid computing," in *Cybernetic Intelligent Systems (CIS), 2010 IEEE 9th International Conference on.* IEEE, pp. 1–6.

[2] Y. Lu, I. Cohen, X. Zhou, and Q. Tian, "Feature selection using principal feature analysis," in *Proceedings of the 15th international conference on Multimedia.* ACM, 2007, pp. 301–304.

[3] M. Maghami, B. Araabi, R. Zoroofi, and M. Shiva, "Facial expression recognition using conspicuous features selection and comparison of the performance of different classifiers," *IEEE International Conference on Signal Processing and Communications, 2007. ICSPC 2007.*

[4] I. Matthews and S. Baker, "Active appearance models revisited," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 135–164, 2004.

[5] T. Cootes, G. Edwards, and C. Taylor, "Active appearance models," *Computer VisionECCV98*, pp. 484–498, 1998.

[6] G. Edwards, T. Cootes, and C. Taylor, "Face recognition using active appearance models," *Computer VisionECCV98*, pp. 581–595, 1998.

[7] G. Edwards, C. Taylor, and T. Cootes, "Interpreting face images using active appearance models," in *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on.* IEEE, 1998, pp. 300–305.

[8] T. Cootes, G. Edwards, and C. Taylor, "Active appearance models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 681–685, 2001.

[9] F. De la Torre, J. Campoy, Z. Ambadar, and J. Cohn, "Temporal segmentation of facial behavior," 2007.

[10] F. Zhou, F. De la Torre, and J. Cohn, "Unsupervised discovery of facial events," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on.* IEEE, 2010, pp. 2574–2581.

[11] M. Zhou, L. Liang, J. Sun, and Y. Wang, "Aam based face tracking with temporal matching and face segmentation," *Proceedings of CVPR'10*, 2010.

[12] T. Cootes and C. Taylor, "Statistical models of appearance for computer vision," Imaging Science and Biomedical Engineering, University of Manchester, Tech. Rep., 2004.

[13] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.

[14] ——, "Equivalence and efficiency of image alignment algorithms," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001.*, vol. 1. IEEE, 2001, pp. I–1090.

[15] G. Fanelli, "Facial features tracking using active appearance models," Master's thesis, Linkpings universitet/Institutionen fr systemteknik, 2006.

[16] C. Chen and C. Wang, "3d active appearance model for aligning faces in 2d images," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2008.* IEEE, pp. 3133–3139.

[17] I. Matthews, J. Xiao, and S. Baker, "2d vs. 3d deformable face models: Representational power, construction, and real-time fitting," *International journal of computer vision*, vol. 75, no. 1, pp. 93–113, 2007.

[18] J. Xiao, S. Baker, I. Matthews, and T. Kanade, "Real-time combined 2d+ 3d active appearance models," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2. Citeseer, 2004.

[19] Y. Sun, M. Reale, and L. Yin, "Recognizing partial facial action units based on 3d dynamic range data for facial expression recognition," in *IEEE 8th International Conference on Automatic Face & Gesture Recognition, FG'08.* IEEE.

[20] M. Cordea, E. Petriu, and D. Petriu, "Three-dimensional head tracking and facial expression recovery using an anthropometric muscle-based active appearance model," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 8, pp. 1578–1588, 2008.

[21] R. Gross, I. Matthews, and S. Baker, "Active appearance models with occlusion," *Image and Vision Computing*, vol. 24, no. 6, pp. 593–604, 2006.

[22] J. M. Buenaposada, E. Muñoz, and L. Baumela, "Efficient illumination independent appearance-based face tracking," *Image Vision Comput.*, 2009.

[23] H. Lee and D. Kim, "Tensor-based active appearance model," *Signal Processing Letters, IEEE*, vol. 15, pp. 565–568, 2008.

[24] R. Gross, I. Matthews, and S. Baker, "Generic vs. person specific active appearance models," *Image and Vision Computing*, vol. 23, no. 12, pp. 1080–1093, 2005.

[25] Y. Cheon and D. Kim, "Natural facial expression recognition using differential-aam and manifold learning," *Pattern Recognition*, vol. 42, no. 7, pp. 1340–1350, 2009.

[26] M. Roberts, T. Cootes, and J. Adams, "Robust active appearance models with iteratively rescaled kernels," in *Proc. British Machine Vision Conference*, vol. 1, 2007, pp. 302–311.

[27] J. Saragih and R. Goecke, "A nonlinear discriminative approach to aam fitting," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on.* IEEE, 2007, pp. 1–8.

[28] S. Zhu and J. Zhao, "Using a robust active appearance model for face processing," in *2009 International Joint Conference on Artificial Intelligence.* IEEE, 2009, pp. 465–468.

[29] Y. Aidarous, S. Le Gallou, A. Sattar, and R. Seguier, "Face alignment using active appearance model optimized by simplex," in *International Conference on Computer Vision Theory and Applications.* Citeseer, 2007.

[30] A. Sattar, Y. Aidarous, S. Le Gallou, and R. Seguier, "Face alignment by 2.5 d active appearance model optimized by simplex," *ICVS, Bielefeld University, Germany*, 2007.

[31] J. Liebelt, J. Xiao, and J. Yang, "Robust aam fitting by fusion of images and disparity data," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2. IEEE, 2006, pp. 2483–2490.

[32] X. Liu, "Generic face alignment using boosted appearance model," in *in Proc. IEEE Computer Vision and Pattern Recognition*, 2007, pp. 1079–1088.

[33] F. Kahraman, M. Gokmen, S. Darkner, and R. Larsen, "An active illumination and appearance (aia) model for face alignment," in *2007 IEEE Conference on Computer Vision and Pattern Recognition.* IEEE, 2007, pp. 1–7.

[34] J. Saragih, S. Lucey, and J. Cohn, "Face alignment through subspace constrained mean-shifts," in *Computer Vision, 2009 IEEE 12th International Conference on.* IEEE, 2009, pp. 1034–1041.

[35] D. Pizarro, J. Peyras, and A. Bartoli, "Light-invariant fitting of active appearance models," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR.* IEEE, 2008.

[36] H. Wu, X. Liu, and G. Doretto, "Face alignment via boosted ranking model," 2008.

[37] Y. Wang, S. Lucey, and J. Cohn, "Enforcing convexity for improved alignment with constrained local models," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on.* IEEE, 2008, pp. 1–8.

[38] A. Sattar, Y. Aidarous, and R. Seguier, "Gagm-aam: a genetic optimization with gaussian mixtures for active appearance models," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on.* IEEE, 2008, pp. 3220–3223.

[39] A. Asthana, J. Saragih, M. Wagner, and R. Goecke, "Evaluating aam fitting methods for facial expression recognition," in *3rd International Conference on Affective Computing and Intelligent Interaction and Workshops, ACII 2009.* IEEE.

[40] X. Liu, "Video-based face model fitting using adaptive active appearance model," *Image and Vision Computing*, vol. 28, no. 7, pp. 1162–1172, 2010.

[41] G. Papandreou and P. Maragos, "Adaptive and constrained algorithms for inverse compositional active appearance model fitting," 2008.

[42] J. Saragih and R. Goecke, "Iterative error bound minimisation for aam alignment," *Pattern Recognition*, vol. 2, pp. 1192–1195, 2006.

[43] M. Storer, P. Roth, M. Urschler, H. Bischof, and J. Birchbauer, "Active appearance model fitting under occlusion using fast-robust pca," in *Proc. International Conference on Computer Vision Theory and Applications (VISAPP)*, vol. 1. Citeseer, 2009, pp. 130–137.

[44] D. Kim, J. Kim, S. Cho, Y. Jang, S. Chung, and B. Kim, "Progressive aam based robust face alignment," in *Proc. of world academy of science, engineering and technology*, vol. 21. Citeseer, 2007, pp. 488–492.

[45] J. Saragih and R. Göcke, "Learning aam fitting through simulation," *Pattern Recognition*, vol. 42, no. 11, pp. 2628–2636, 2009.

[46] T. Cootes and C. Taylor, "An algorithm for tuning an active appearance model to new data," in *Proc. British Machine Vision Conference*, vol. 3. Sl]:[sn], 2006, pp. 919–928.

[47] J. Daugman, "High confidence visual recognition of persons by a test of statistical independence," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 15, no. 11, pp. 1148–1161, 1993.

[48] M. Lades, J. Vorbruggen, J. Buhmann, J. Lange, C. von der Malsburg, R. Wurtz, and W. Konen, "Distortion invariant object recognition in the dynamic link architecture," *Computers, IEEE Transactions on*, vol. 42, no. 3, pp. 300–311, 1993.

[49] N. Petkov, "Biologically motivated computationally intensive approaches to image pattern recognition," *Future Generation Computer Systems*, vol. 11, no. 4-5, pp. 451–465, 1995.

[50] N. Campbell and B. Thomas, "Automatic selection of gabor filters for pixel classification," in *Image Processing and Its Applications, 1997., Sixth International Conference on*, vol. 2. IET, 1997, pp. 761–765.

[51] L. Shen and L. Bai, "Adaboost gabor feature selection for classification," in *Proc. of Image and Vision Computing NewZealand.* Citeseer, 2004, pp. 77–83.

[52] S. Li and Z. Zhang, "Floatboost learning and statistical face detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 9, pp. 1112–1123, 2004.

[53] G. Tourassi, E. Frederick, M. Markey, and C. Floyd Jr, "Application of the mutual information criterion for feature selection in computer-aided diagnosis," *Medical Physics*, vol. 28, p. 2394, 2001.

[54] F. Fleuret, "Fast binary feature selection with conditional mutual information," *The Journal of Machine Learning Research*, vol. 5, pp. 1531–1555, 2004.

[55] A. Pentland, B. Moghaddam, and T. Starner, "View-based and modular eigenspaces for face recognition," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on.* IEEE, 1994, pp. 84–91.

[56] Y. Freund and R. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational learning theory.* Springer, 1995, pp. 23–37.

[57] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," 2001.

[58] M. Zhou and H. Wei, "Constructing weak learner and performance evaluation in adaboost," in *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on.* IEEE, pp. 1–4.

[59] S. Gallant, "Perceptron-based learning algorithms," *Neural Networks, IEEE Transactions on*, vol. 1, no. 2, pp. 179–191, 1990.

[60] N. Otsu, "A tlreshold selection method from gray-level histograms," *Automatica*, vol. 11, pp. 285–296, 1975.

[61] D. Tax and R. Duin, "Using two-class classifiers for multiclass classification," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 2. IEEE, 2002, pp. 124–127.

[62] G. McCabe, "Principal variables," *Technometrics*, pp. 137–144, 1984.

[63] I. Jolliffe, "Principal component analysis," *Encyclopedia of Statistics in Behavioral Science*, 2002.

[64] W. Krzanowski, "Selection of variables to preserve multivariate data structure, using principal components," *Applied Statistics*, pp. 22–33, 1987.

[65] ——, "A stopping rule for structure-preserving variable selection," *Statistics and Computing*, vol. 6, no. 1, pp. 51–56, 1996.

[66] L. van der Maaten, “Basic matlab implementation of the aam,” Online available on: http://homepage.tudelft.nl/19j49/Active_appearance_models.html.

[67] M. Stegmann, B. Ersboll, and R. Larsen, “Fame-a flexible appearance modeling environment,” *IEEE Transactions on Medical Imaging,*, vol. 22, no. 10, pp. 1319–1331, 2003.

[68] R. Stapenhurst, “A demo to illustrate the behaviour of adaboost,” Online available on: http://www.mathworks.com/matlabcentral/fileexchange/29245-boosting-demo&watching=29245.

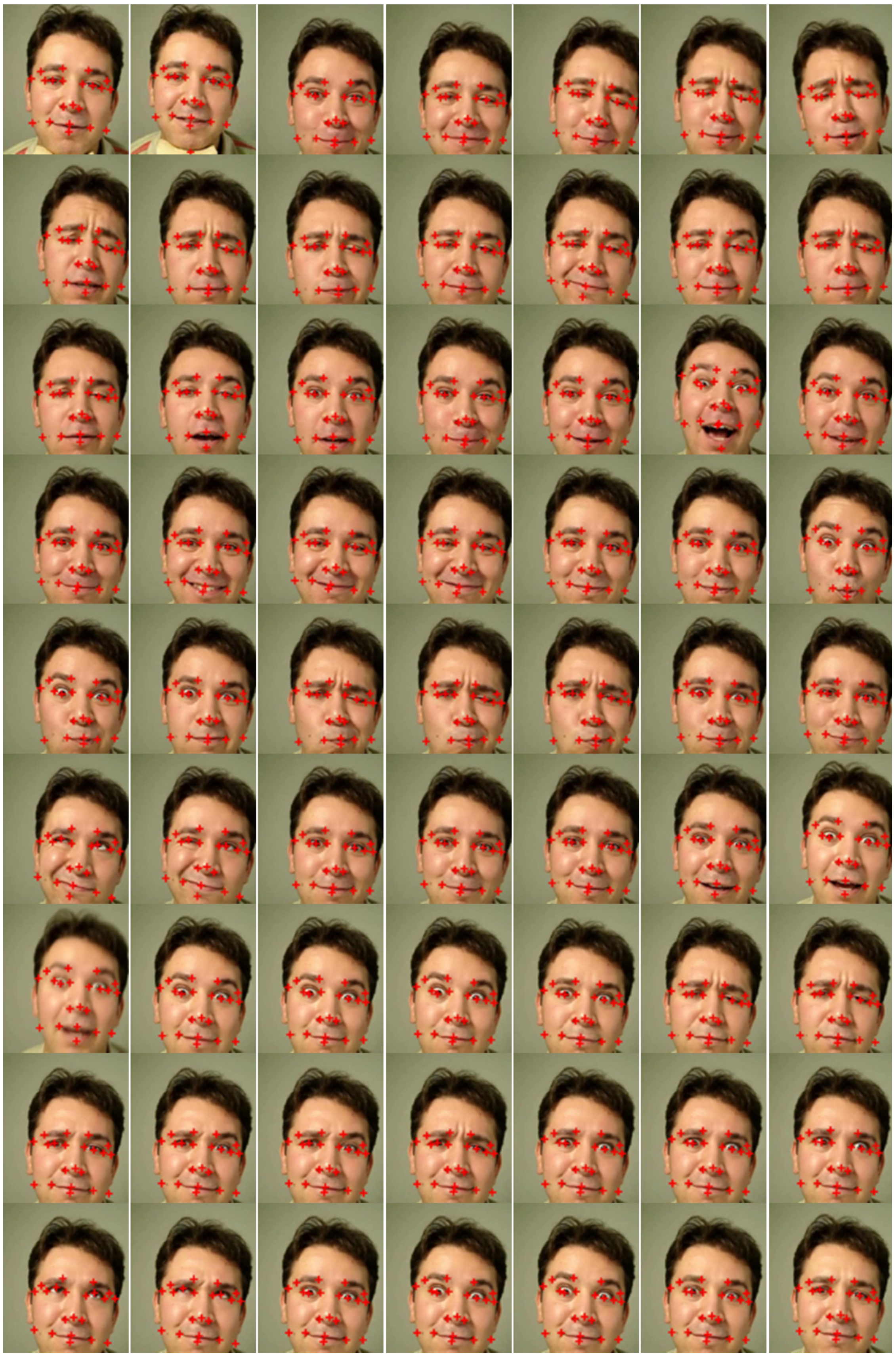


Figure 20: Facial feature tracking by AAM.