Chapter 4

Classification concepts

4.1 Introduction

The purpose of the classifier is assigning a label (in this case "depressed" or "non-depressed") to a data sample (eg. a human face). Automatic classification can usually be split into two different phases, training and testing.

The first phase is the training of the classifier, where a set of labeled data samples (the "training set") is used to adjust the internal parameters of the underlying classification algorithm. Once the training phase is complete, the algorithm with the adjusted parameters can be used on new data samples to estimate or predict the corresponding (theoretically unknown) class label. This latter set is called the "testing set" in classification literature. It is common for the correct labels to be known for both data-sets, being the testing set labels only used to assess classification accuracy [4], although there are cases in which the classifier is used for labeling completely unknown data.

For two-class classification problems, the training set is usually expressed as:

$$\{(\boldsymbol{x}_1, y_2), \ldots, (\boldsymbol{x}_p, y_p)\} \tag{4.1}$$

Where $\boldsymbol{x}_i \in \Re^n$ is the feature vector of a particular sample to be classified, and $y_i \in \{-1, 1\}$ is the corresponding label, indicating class membership as a function of its sign.

At least three crucial design choices can be distinguished: the choosing of adequate training examples, the choosing of a representation for the data sample, and the choosing of the classification algorithm. These three important points will be discussed in the following sections.

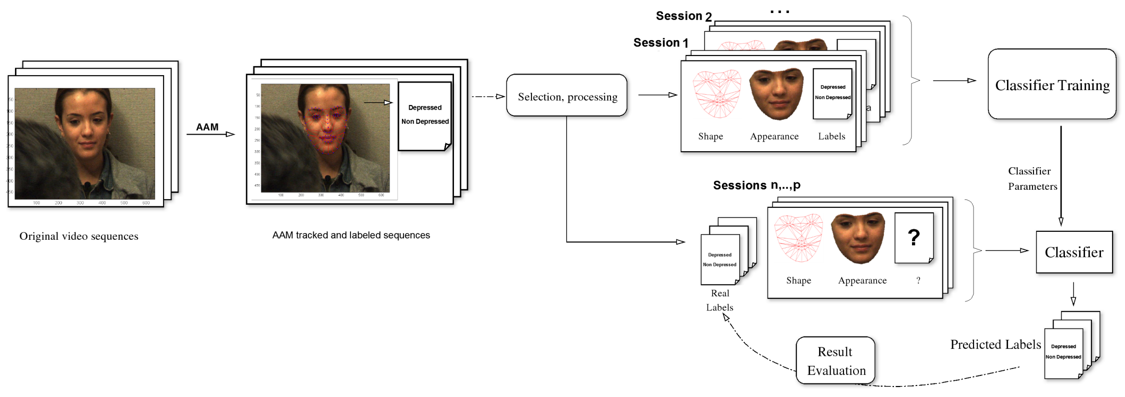

Figure 4.1: Classification overview.

4.1.1 Training set

In the training phase, the classifier "learns" its internal parameters based on the input, which is a set of features extracted from a certain set of videos. For each video sequence chosen as part of the training set, a set of features will be extracted. These features will have a label associated, which in this case will mean if the features themselves correspond to a depressed or non-depressed example.

In this particular case, the labeling of the videos is a very easy task. The depressed or non-depressed status of the a patient in a given session is only related to the Hamilton score associated to it (see Appendix A). However, there are other cases in which this labeling process is very costly, both in time and resources, for instance, when it is necessary to annotate the Action Units occurrences using the Facial Action Coding System [14].

Once the video sequences have been labeled, good training examples must be chosen. Examples that are too subtle (not noticeable in the extracted features) or otherwise noisy will only "confuse" the classifier's training algorithm, resulting in worse parameters than if they had been excluded. This has lead us to choose sessions where the patients are either very depressed or non-depressed (remitted subjects), which reflects in the criterion adopted, explained in Section 2.2.3.

4.1.2 Data sample representation

The data sample to be classified (conceptually, a face) must be represented in a way the classifying algorithm can handle. This is usually a real valued multidimensional vector $\boldsymbol{x}_i \in \Re^n$, the components of which are the features determined to carry discriminative information.

The problem is more complex than might appear. Since the number of examples available for each class is limited, the chosen features might confuse the classifier if the sampling of them that has been made in the training set is not representative of the population of samples at large [8].

Therefore, one cannot input all imaginable features related to the problem at hand to the classifier and expect good results. A previous feature selection step must be carried out. This selection can be, strictly speaking, a selection (some of the generated features are included, others are left out), or a transformation of the original set.

There are a number of feature selection algorithms available. Most are based on calculating some measure (eg. correlation, mutual information, classification performance considering only this feature, etc.) between features and class labels and selecting features given a threshold value for this measure, or on a transformation of the original feature space which preserves only the discriminative information [1].

The transformations applied to the features are typically linear transformations, for simplicity. The most popular methods are component analysis methods, such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) and variations. The transformation can usually be expressed as a projection into a different basis, $\tilde{\boldsymbol{x}} = \mathbf{B}^T \cdot \boldsymbol{x}$ with $\mathbf{B} \in \Re^{n \times n'}$ transforming the n dimensional feature space into n', with $n' < n$ usually. In PCA, the matrix $\mathbf{B}$ minimizes the reconstruction error for the training data (see Section 3.3.2). In LDA, $\mathbf{B}$ maximizes the variance between classes while minimizing within class variance. If the values of $\mathbf{B}$ are limited to $\{0, 1\}$, with one non-zero value per column, a discrete feature selection is obtained [6].

4.1.3 Choosing the classifier

Classification results have been published with almost all available classification algorithms [17, 26]. The results obtained (and the applicability of each algorithm) depends on the nature of the data to classify, which in this case depends on the features chosen, and the video recording conditions of the database. Good results have been reported on different data-sets (and different feature sets), especially in the context of classification based on AAM tracked video sequences, with a Support Vector Machine [21, 22], where results with more than 90% accuracy where reported on the Cohn-Kanade posed expressions database. The SVMs are state-of-the-art classification tools and are considered to be among the most versatile and efficient classification algorithms. SVMs will be, therefore, the algorithm of focus of this project.

4.1.4 Cross validation

Once the classifier is chosen, a data representation has been found, and labeled examples have been collected, the classifying and feature selection algorithms must be trained and subsequently tested.

It is often the case that one does not have enough data available to do as thorough an evaluation of the performance as one would wish (data collection is often costly). One common way to mitigate the effect of having a small data-set and providing a more reliable evaluation of the methods used is cross-validation. This is a training and testing scheme where the training-testing cycle is run multiple times on different subsets of the complete labeled data-set available to the researcher.

The general procedure is called K-fold cross-validation, which involves partitioning the complete data-set into K subsets. In each training-testing cycle, one subset is used for testing while the remaining K-1 subsets are used as training information. The procedure is repeated so that all the samples are found once in the testing set. The subset size is chosen so that enough samples are available for the training step and testing set in each cycle. Statistics of the performance across training-testing cycles, such as mean error, can then be used to evaluate performance. Hyper-parameters of the classification (such as the SVM C parameter, or the Gaussian kernel σ, discussed in the next section) are also often tuned using a cross-validation procedure [4].

In the case K is chosen to be the total numbers of samples available in the database, in each of the cycles one single sample is used for the testing, while the remaining ones are used for the training phase. This is the so-called *leave one out* method.

4.2 SVM classification

4.2.1 Overview

SVMs are a linear (during testing) classification method which (during training) searches for the maximum margin hyper-plane (in the feature space) which best separates the samples to be classified. The criteria to evaluate this “best” separation is due to Vapnik and Cortes [12], based on previous work on optimal hyper-plane classification from Vapnik: the idea is to maximize the distance to the hyper-plane (the margin) of each of the closest samples of each class. Vapnik and Cortes introduce a penalization term ξ_i for misclassified samples, for the non-separable case.

$$\begin{aligned} \min \frac{\|\boldsymbol{w}\|}{2} + C\sum_{i=1}^{p}\xi_i \qquad & \textit{such that} \\ y_i(\boldsymbol{w}\cdot\boldsymbol{x_i}+b) \geq 1-\xi_i, \qquad & i=1,\ldots,p \\ \xi_i \geq 0, \qquad & i=1,\ldots,p \end{aligned} \tag{4.2}$$

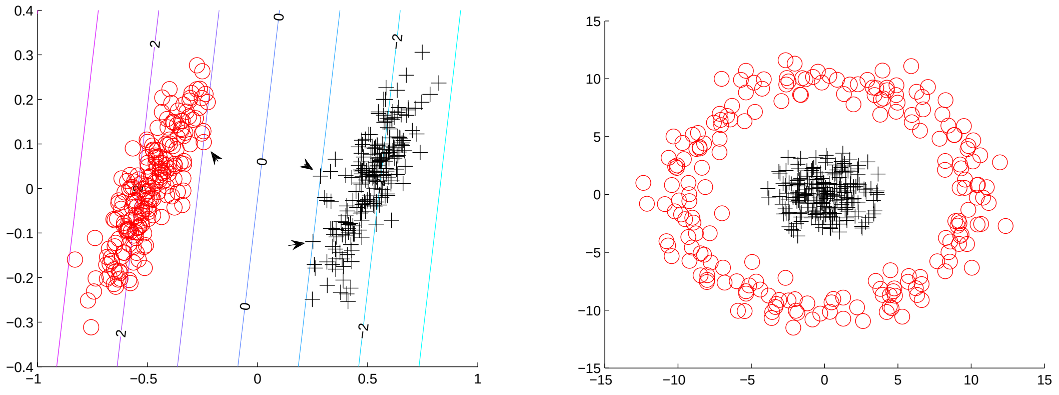

Figure 4.2: SVM two class linear classification example on a 2D data-set. Left: The contours show the margin (distance to the hyperplane, at 0). The arrows show the support vectors. Right: Example of a non-linearly separable 2D data-set.

The parameters $\boldsymbol{w}$ and b, are determined by a quadratic programming optimization on the training data. The parameter C (considered a hyper-parameter, i.e. a parameter of the training process itself) must usually be determined empirically for the data-set in question (typically, by some trial-and-error optimization scheme). The LibSVM implementation [5] was used for all experiments.

4.2.2 Kernel methods

The linear separability restriction of this algorithm is usually overcome by transforming the original feature space into a higher dimensional feature space where the classes can (hopefully) be separated by a hyper-plane. This can be done efficiently by employing the so-called "Kernel Trick": substituting dot products in the algorithm's formulation with adequate kernel functions (generalized dot product functions in a Hilbert space) [12].

As an intuitive explanation, a polynomial kernel will be used to separate the example data-set (right) in Figure 4.2. The data-set is arranged in concentric circles, where the distance to the origin is the most discriminant feature. One higher dimensional mapping where this data is linearly separable is the 3D space where each 2D point $\langle u, v\rangle$ is mapped to $\langle u^2, v^2, uv\rangle$. Since circles have $u^2 + v^2 = c$, concentric circles describe lines in the first two coordinates of the new higher dimensional (3D, in this case) space. This mapping is shown in Fig. 4.3 (left). In this space, a plane that separates the two classes can easily be found.

For the purposes of actual computation, instead of mapping the data to a higher dimensional space, the kernel trick involves finding an efficient expression for the dot product in this higher dimensional space, preferably in terms of the original components. The higher dimensional dot product in this example is:

$$\left\langle u_1^2, v_1^2, u_1v_1\right\rangle \left\langle u_2^2, v_2^2, u_2v_2\right\rangle = (u_1u_2)^2 + (v_1v_2)^2 + 2u_1v_1u_2v_2 = (\langle u_1, v_1\rangle \langle u_2, v_2\rangle)^2 \tag{4.3}$$

namely, the squared dot product in the original 2D space. This is called the kernel function for this mapping; in this case, a polynomial kernel of degree 2. By expressing all computations of the SVM algorithm in terms of dot products of features, one can incorporate any higher dimensional mapping by substituting the dot products with an adequate kernel function.

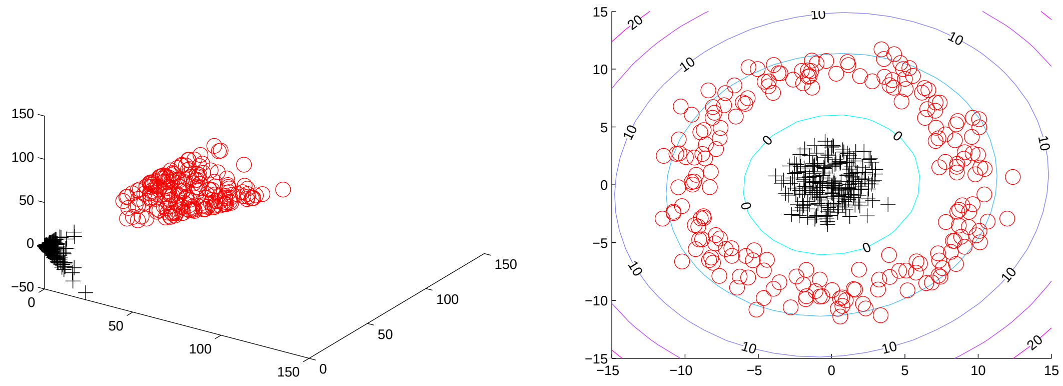

Figure 4.3: Polynomial SVM classification example on a 2D data-set. Left: The original 2D data-set mapped to 3D. Right: The margin separation contours (distance to the 3D optimal separating plane) reflected in the original 2D space.

Finding the kernel function that allows for linear separation of a particular data-set is not trivial, and proving that a certain function corresponds to a dot product in some space is difficult too. Normally, only a set of the most popular kernels choices is tried.

Among these, the exponential kernel (sometimes called Gaussian or radial basis function kernel) is perhaps one of the most popular, where the kernel function is:

$$K(\boldsymbol{s}_1, \boldsymbol{s}_2) = \exp\left(-\frac{\|\boldsymbol{s}_1 - \boldsymbol{s}_2\|^2}{2\sigma^2}\right) \tag{4.4}$$

As can be seen in Eq. (4.4), the similarity measure induced by the kernel depends on distance and the influence of the support vectors is modified by the parameter σ. This parameter must usually be determined empirically.

4.3 Performance measures

4.3.1 Performance parameters

As mentioned in previous sections, hyper-parameters of the classification such as the SVM C parameter, or the Gaussian kernel σ, are often tuned using a cross-validation procedure [4]. Basically, the pair of C and σ which provide the best results during the cross validation are chosen. The goodness of the result is measured in the experiments by the accuracy, which is computed by taking into account the following set of parameters: the true and false positive rates and the true and false negative rates. The definition of all these terms is presented above:

- **True positive rate (TPR)** is the number of positives examples that have been correctly classified over the total number of positive samples.
- **False negative rate (FNR)** is the number of positive examples that have been incorrectly classified over the total number of positive samples.
- **True negative rate (TNR)** is the number of negative examples that have been correctly classified over the total number of negative samples.
- **False negative rate (FNR)** is the number of negative examples that have been incorrectly classified over the total number of negative samples.
- **Accuracy** in binary classification is the number of true positives plus the number of true negatives over the total number of samples.

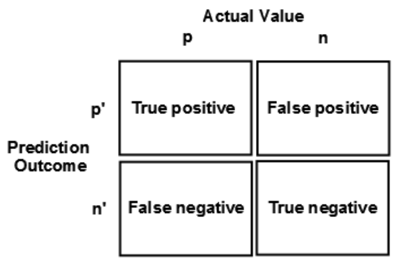

Figure 4.4: Confusion matrix.

In addition to the above mentioned measures, another parameter is used as a complement of the accuracy for the evaluation of the classification performance. This is the parameter F, and it is a function of the precision and recall measures.

$$\text{precision} = \frac{TPR}{TPR + FPR} \tag{4.5}$$

$$\text{recall} = \frac{TPR}{TPR + FNR} \tag{4.6}$$

$$F = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{4.7}$$

4.3.2 ROC curves

The parameters described in the previous section depend on the threshold value used for classification, given by the SVM. To provide some independence from this value, Receiver Operating Characteristic (ROC) curves are often used to display classification results. This curve displays the False Positive Rate (FPR) against the True Positive Rate (TPR), while sweeping all the possible values for the threshold of the SVM.

Some examples of ROC curves are shown in Figure 4.3.2. The *y-axis* corresponds to the TPR and the *x-axis* to the FPR. The sweeping begins in the right hand side of the curve, setting the threshold lower than any output of the SVM. In this scenario, every sample is classified to belong to the positive class (depressed) and the TPR and FPR take their maximum value, 1. On the other hand, the sweeping ends when the threshold is set to be higher than any output of the SVM, being all the samples classified as negative (non-depressed) and the TPR and FPR taking their minimum value, 0. The threshold chosen for the SVM as the optimum value, in other words, the value that gives us the maximum accuracy, can be visualized in the figure as the point that maximizes the values of the pair (TPR, FPR).

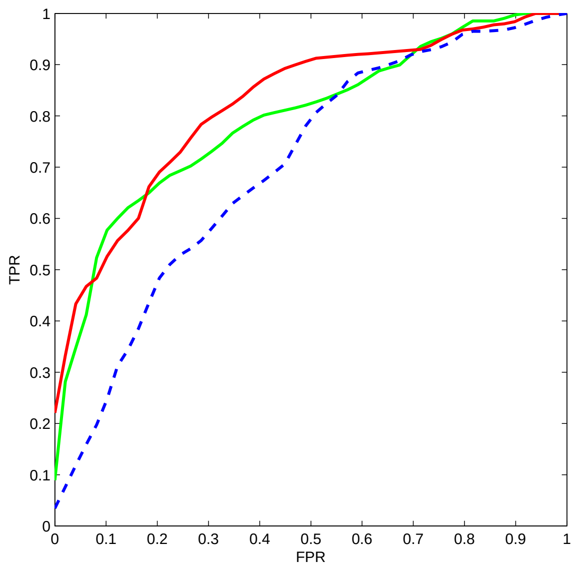

Figure 4.5: Examples of ROC curves.

A measure extracted from this curve is the area under it and its value is used for measuring the performance of the classification experiment. An ideal curve has area 1.

4.4 Normalization

4.4.1 Overview

A perfect tracking of the human face does not suffice. Since we are trying to build a person independent classifier (or, as person independent as possible under the constraint of limited training data) the correspondence between tracked features in consecutive frames must be dealt with accurately. This is the reason why a normalization is needed previous to the feature extraction procedure. Correctly aligning, scaling, and rotating the shapes is what is meant by normalization.

One of the most important challenges of the normalization method is to be able to separate the rigid and non-rigid components of the movement. While the rigid deformation of the landmark points is of little interest (at least, in a first approximation), the non-rigid deformations of the feature points will carry the bulk of the discriminant information that will allow us to separate between the depressed and non-depressed class of subjects. As such, it is crucial to our purpose that the facial normalization compensates as much of the rigid deformation (that is, relative head position, rotation and gross scale [29]) as possible, while maintaining the non-rigid deformations (the shape of the mouth, the position of the eyebrows, etc.) intact. These motions are intimately coupled in the movement of the tracked points, and separating them is not simple.

In a two dimensional alignment framework, it is common to model the rigid transformation as an affine or similarity transformation applied to the non-rigid deformation basis of the AAM model. The AAM training and fitting process itself typically incorporates a similarity transformation precisely meant to separate these motions. While it is tempting to attempt classification based on the AAM parameters directly (most of the rigid motion has been compensated for in the fitting process and has shown promising results [22, 21]), the process of re-aligning the faces will be tackled separately from the AAM fitting algorithm for several reasons. Firstly, the AAM fitting algorithm minimizes a measure of error based on the likeness of the modeled face to the actual video frame. Thus, the error function minimized does not necessarily satisfy the requirements for classification, but only a goodness of fit of the AAM model. Secondly, it depends entirely on the trained AAM model. Depending on how the model was created, the shape variation basis will or will not incorporate a certain amount of rigid variations which will be confused with non-rigid motion. Thirdly, not using the AAM parameters directly gives a measure of independence from the underlying tracking algorithm. Conceptually, a different tracker could be used without drastically affecting the rest of the system.

In the following sections two alignment methods will be reviewed and compared: the similarity transformation and the affine transformation.

4.4.2 Similarity transformation

The similarity transform is the simplest to be considered. It has four (4) parameters to be optimized: scale, rotation, and two translational components. The deformation it produces is minimal in comparison with other methods considered. This can be both positive and negative: it does not excessively deform the shape of the face (therefore preserving most of the information), but conversely, its descriptive power is limited and cannot compensate for as much out of plane head motion as other methods.

In the following, $\boldsymbol{x}_i$ will be a position vector for a landmark point i in homogeneous coordinates (padded with one to $\Re^3$) and $\mathbf{X}_j \in \Re^{3\times n}$ will be a matrix with $\boldsymbol{x}_i$ as columns for the shape at frame j. The vector $\boldsymbol{p}$ will symbolize the underlying parameters of the transformation used.

$$\boldsymbol{T}_s(\mathbf{p}) = \begin{bmatrix} a & b & tx \\ -b & a & ty \\ 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

4.4.3 Affine transformation

The affine transform can ideally compensate for the rigid motion of points on a plane undergoing a perspective projection. As most of the points of interest in the face can be said to lie on a plane (when one considers relative distance to the camera), this transformation would be ideally suited to our purposes. When applying this transformation, more care must be taken because it can introduce unwanted deformations.

$$\boldsymbol{T}_a(\mathbf{p}) = \begin{bmatrix} a & b & tx \\ c & d & ty \\ 0 & 0 & 1 \end{bmatrix} \tag{4.9}$$

4.4.3.1 Iterative alignment algorithm

To align a set of shapes, an iterative scheme is used. When the similarity transform is considered, this is usually called iterative Procrustes alignment [28]. $\tilde{\mathbf{X}}$ denotes a transformed shape. The shapes are aligned to a mean shape, which is recalculated with the transformed shapes at each iteration step, where:

$$\tilde{\mathbf{X}}_j^{iter} = \boldsymbol{T}(\mathbf{p}_j)\,\mathbf{X}_j, \qquad \textit{such that,} \qquad \mathbf{p}_j = \arg\min E\left(\mathbf{X}_j; \mathbf{p}_j; \left\{\tilde{\mathbf{X}}_{1,..,n}^{iter-1}\right\}\right) \tag{4.10}$$

Several error measures $E\left(\mathbf{X}_j; \mathbf{p}_j; \left\{\tilde{\mathbf{X}}_{1,..,n}^{iter-1}\right\}\right)$ for the alignment of a shape can be chosen. The most common by far is the sum of the L2 norm difference between the coordinates of the mean shape of the previous iteration and the current warped shape:

$$E\left(\mathbf{X}_j; \mathbf{p}_j; \left\{\tilde{\mathbf{X}}_{1,..,n}^{iter-1}\right\}\right) = \left\|\mathbf{W} \odot \left(\boldsymbol{T}(\mathbf{p}_j)\,\mathbf{X}_j - \bar{\tilde{X}}^{iter-1}\right)\right\|_F \tag{4.11}$$

In Figure 4.6 a point cloud corresponding to the faces of a single session is shown, before and after the alignment process.

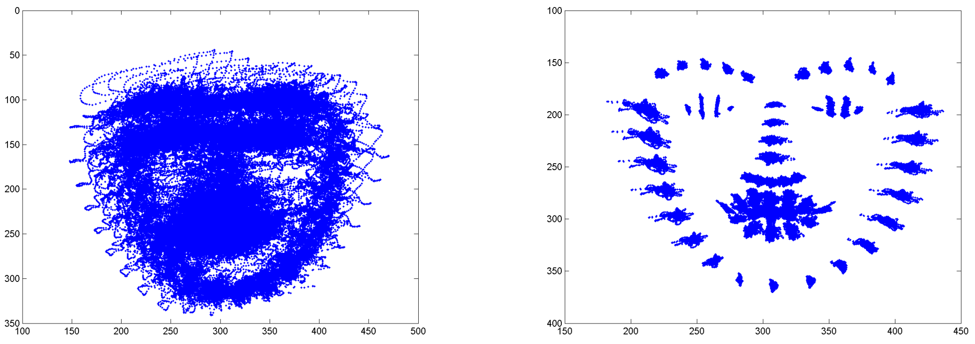

Figure 4.6: Left: Unaligned point cloud. Right: Similarity transform aligned point cloud.

4.4.4 Inter-subject variability and global mean shape

There is great variability between human faces, especially in size (also due to distance from the camera) but also in shape. The question arises whether some explicit normalization of the facial features should be attempted, or whether this should be left up to the classifier.

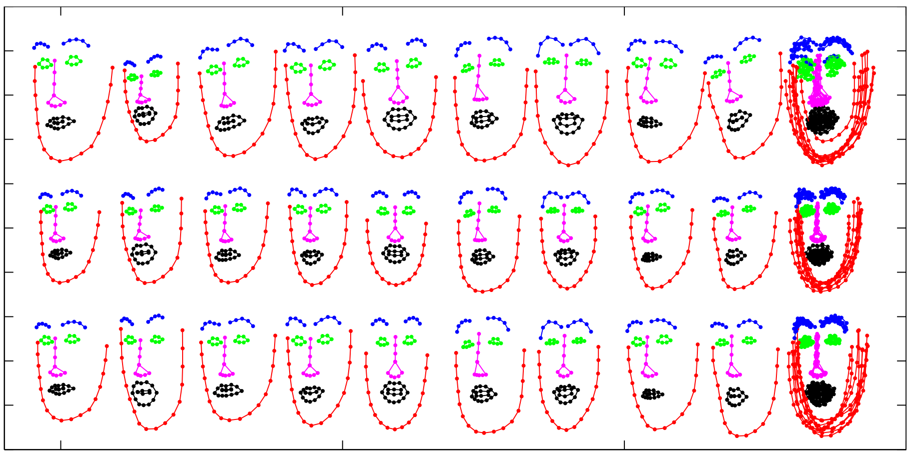

Figure 4.7: First frame of a sequence for 9 Spectrum subjects. Top: Zero centered faces of the original sequences. Middle: Similarity aligned using most rigid points (nose, eye corners). Bottom: Affine aligned using most rigid points. Right: Superimposed shapes.

If iterative Procrustes analysis is applied, every session will be aligned with high accuracy to its own shape (the process will minimize a function of error), but the inter-subject variability will be as well high. In the matter at hand, it is essential to be able to compare the features extracted from different subjects. Therefore, and although it will lead to a lower accuracy in the alignment process, it is tempting to apply an inter-subject normalization method to the subjects of the Spectrum database.

One of the simplest ways to address this problem is to align every session of the database with respect to a common mean shape. For a better alignment, it has to be a mean shape that satisfies the relative position of the subject and the camera itself. When the Spectrum database was recorded, two cameras were placed in both sides of the subject (the idea was to be able to perform a three dimensional tracking) but finally just the left one was used for the tracking. Therefore, the pose of the subject is not frontal: him/her left hand side of the face is more visible. This is the reason why, when choosing a common mean face for the alignment it is recommended to take one that fulfils this condition.

A good example of a mean shape with the characteristics discussed above is the mean face resulting of the averaging of all the faces of the database. That is, 149 sessions with an average length of 20,000 frames. In the following, this mean shape will be referred to as *global mean shape*.

Once the mean face used is chosen, it is time to think which of the two methods of alignment discussed would suite best our goal. As we are dealing with faces and we want the alignment process to cause as small deformation as possible, the similarity transformation is a good option. The drawback of using the global mean shape is that no iterative method could be applied, and therefore no function of error could be minimized. In sum, a single similarity transform is applied to each frame, which aligns it with respect to the global mean face.

In Figure 4.8 a comparison between the alignment of a face with respect to its session mean shape and the alignment with respect to the global mean shape is shown. As it can be seen in the figure, the first one aligns each of the frames of a session with respect to its own mean shape very precisely. Nevertheless, since after this alignment the frames belonging to different sessions are not aligned between them and thus cannot be compared, it does not provide a good framework for a feature extraction procedure. The opposite could be said about the alignment with respect to the global mean shape. Although it is not as precise as the aforementioned case within a session, the shapes of different sessions are transformed to be as close to a common shape. This means that when the feature extraction method is applied, the features belonging to different subjects/sessions can be compared to each other, making possible the classification into the depressed and non depressed class.

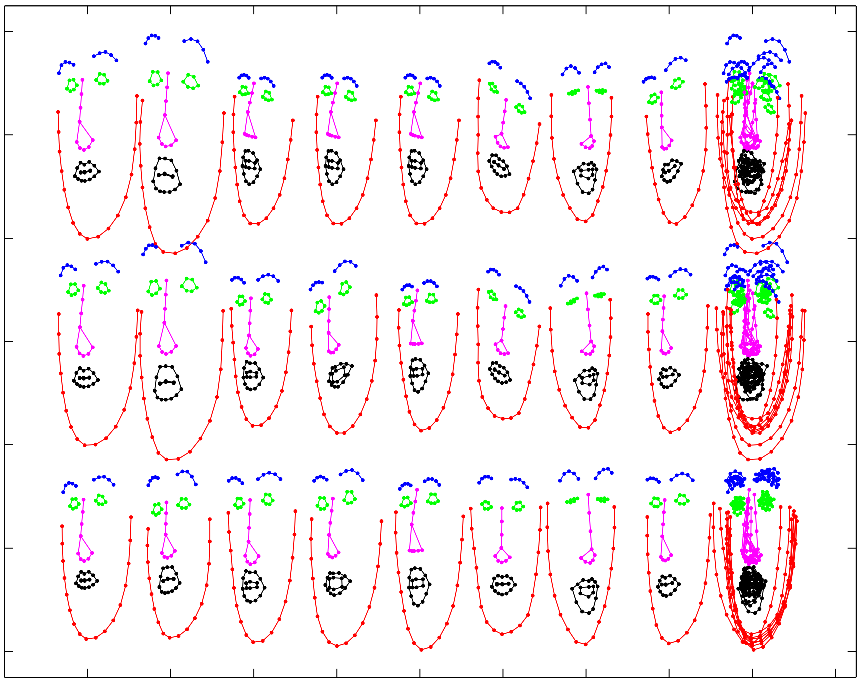

Figure 4.8: 77th frame of a sequence for 8 Spectrum subjects. Top: Zero centered faces of the original sequences. Middle: Similarity aligned using its corresponding mean shape. Bottom: Similarity aligned using the global mean shape. Right: Superimposed shapes.