IDIAP RESEARCH REPORT

Audio- Video Person Clustering In Video Databases

Frederic Kottelat [a], Jean-Marc Odobez [b]

IDIAP–RR 03-46

15. September 2003

Dalle Molle Institute for Perceptual Artificial Intelligence • P.O.Box 592 • Martigny • Valais • Switzerland

phone +41 − 27 − 721 77 11
fax +41 − 27 − 721 77 12
e-mail secretariat@idiap.ch
internet http://www.idiap.ch

a EPFL, Eurecom, frederic.kottelat@eurecom.fr
b IDIAP, odobez@idiap.ch

IDIAP Research Report 03-46

Audio- Video Person Clustering In Video Databases

Frederic Kottelat, Jean-Marc Odobez

15. September 2003

Contents

Abstract *A methodology towards person clustering in meeting databases is presented in this report. Such a goal is generic to a number of problems in computer vision and more specifically in content-based video indexing and retrieval applications.*

First, the audio-stream was considered alone, leading to the speaker clustering problem. An already existing algorithm has been used to this end. Then, the video stream was analysed, leading to a face clustering algorithm build from probability density distribution of face similarity distances. Finally, both modalities are considered to combine voice clustering and face clustering and achieve person clustering.

This method has been tested on the IDIAP meeting database, and many results are given to prove the efficiency of the method and to show that it can be applied to other databases.

Résumé *Une méthode de détection du nombre de personnes présentes dans des fichiers vidéo a été développée en combinant les résultats obtenus par regroupement de voix semblables d'une part, et regroupement de mêmes visages d'autre part. Le problème de détection (de visages, de voix ou de personnes) est un problème majeur pour l'amélioration des indexations et des recherches de vidéo.*

La première tâche consistait à regrouper les voix semblables en utilisant un algorithme existant. L'effort le plus important a porté sur le problème de classification de visages: calcul de mesures de similarité entre visages et afin de définir lesquels sont identiques et lesquels ne le sont pas. Enfin la dernière partie consistait à fusionner les résultats obtenus sur les voix et sur les visages.

Cette méthode a été testée sur une base de donnée de réunions enregistrées dans les locaux de l'IDIAP et pourrait être généralisée à d'autre bases de données.

Chapter 1

Introduction

Digital video libraries are generating tremendous interest in pattern recognition, computer vision, and multimedia research communities. Powerful processors, high-speed networking, high-capacity storage devices, improvements in compression algorithms, and advance in processing of audio, speech, image, and video signals are making libraries technically feasible.

Content-based video indexing and retrieval is an active research topics due to the enormous amount of *unstructured* data of these libraries, the spread of its use as a data source in many applications and the increasing difficulty in its manipulation and retrieval of the material of interest. The need for content-based indexing and coding has been foreseen by ISO/MPEG that introduced two new coding standards: MPEG-4 and MPEG-7.

The first step in this direction has been the automatic extraction of video structures (*summaries*). However objects, and in particular people, correspond to the desired level of access to a video database. With this in mind, the generation of automatic person-based video structures constitutes a valuable feature that complements the video summary representation. When no model of already known people is available, the problem can be seen as one of **person clustering**. This is a common problem in real databases, where annotation/data collection for building identities of all possible participants in a video is inconvenient or not possible.

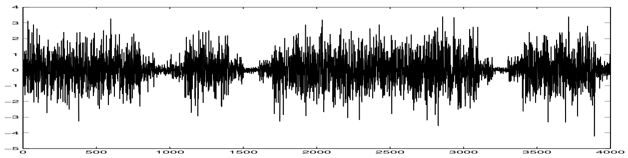
1.1 Goal of the project

A first statement of the project is the extraction from a database of audio-visual data streams, the detection of all the people appearing in the video and the identification of all the audio-visual sequences in which they are speaking. Identification of segments could be achieved by a vision-based speaker detection in association with a *speech* segmentation. On the other hand, speech segment clustering and face image clustering can lead to the detection of the number of people. Next figure (1.1) shows a video sequence in which are three persons, and the corresponding audio segment. The association between audio sequences and video sequences is one of the major problem. In this example, we can for instance assume that the girl appearing in the two first images is the person speaking at the beginning of the audio segment, but who is speaking in the third image? Comparisons with further audio features can maybe solve this problem.

In this project I focus on meetings database recorded at IDIAP. A method is developed in order to find how many participants are involved in the whole database by combining voice clustering and face clustering.

Face clustering is done by first extracting as many faces as possible and then computing reference images of each participant in every meeting and secondly by clustering all the *reference* images found based on face similarity distances. An existing algorithm of audio clustering has been modified in order to deal with our problem of voice clustering.

a



b

Figure 1.1: Example of audio-visual sequence: (a) video sequence, (b) audio sequence

1.2 Plan of the report

Before going deeply through the project a presentation of the existing material and databases is presented in chapter 2. Chapter 3 presents a survey of clustering. Then next chapters treat more specifically the theory and research activities. Chapter 4 describes in detail the work on audio clustering and shows results. A description of a face displacement estimation algorithm and the modifications provided are given in chapter 5: this algorithm is used to compute a (dis)similarity between two faces. The biggest part of the project is described in chapter 6 on *Faces Clustering*. The studied methods to achieve the goals are explained and commented on. Chapter 7 describes how are merged audio and video clustering and shows the project results. Finally a chapter concludes this report by defining possible future works and analyzing all results. Annexe A gives comments on vision-based speaker detection and the reason why this subject has not been used in further researches.

Chapter 2

The Database

2.1 Introduction

A small presentation of the databases that I used during all my internship is given in this section. First a special meeting room used for recording audio-visual databases is described. The main database and other advanced data are described in the next sections of this chapter. Further researches are only based on this database and the advanced data described later will be very useful, especially for the audio clustering.

2.2 Smart Meeting Room

The IDIAP Smart Meeting Room [1] is a meeting room equipped with multi-channel audio-visual recording facilities. It was installed for the purpose of acquiring audio, visual and textual data within meeting scenarios. These recordings are needed to support the wide range of speech, image and multi-modal research efforts that occur at IDIAP, and at other partner institutions.

The Smart Meeting Room (see 2.1) is a 8.2m x 3.6m x 2.4m rectangular room containing a centrally located rectangular table (suitable for seating 12 people). A white-board and retractable projector screen occupy the wall at one end of the room, while the audio-visual acquisition equipment is at the other end. Metal rails have been installed on all outer edges of the ceiling to allow flexible cameras and lights.

The Smart Meeting Room has 24 miniature lapel microphones, which can be used either as a lapel microphone attached to a meeting participant, or as part of tabletop microphone arrays. Moreover it currently has three video cameras that can be placed at any location around the room.

Thus, the IDIAP Smart Meeting Room is capable of recording high quality, multi-channel, audio-visual data. The current configuration for meeting data uses three cameras, six lapel microphones, and two 8-element microphone arrays to record meetings containing up to 6 people.

2.3 Meeting data

The main IDIAP database consists of two series (a for the train sets and the other for the test sets) of 30 fake meetings which have been recorded at the IDIAP Smart Meeting Room. It is an important and very useful part of the IM2 projects (Interactive Multimodal Information Management) and all the data are available for IDIAP's partner institutions from the website http://mmm.idiap.ch. (see 2.2).

The choice of recording this database has been motivated by the fact that many projects at IDIAP deal for instance with projects like notes-taking detection, face or skin detection, head- or arm-motion

Figure 2.1: The IDIAP Smart Meeting Room

detection in image processing and many others in speech processing. The recording conditions are described below.

There are seven different participants in each class, and four people participate in each meeting. The topics of these meetings are very simple, here are some examples:

The last book I have read, *The last movie I have seen*, *The plan for my next holidays*.

A "scripted meeting" approach was taken to collect the required audio-visual data for the meeting action recognition experiments. A set of legal meeting actions was defined as:

monologue, note-taking, presentations, consensus, disagreement, discussion.

Three different views have been recorded: the first camera shows two meeting participants, the second camera shows the two other people and the third camera is pointed the white-board and projector screen out (see 2.3).

Concerning the audio, twelve files per meeting have been archived. Four of them are the lapel microphones attached to each participant, the remaining files are the 8-element microphone array. Each files are stored at a sampling rate of 8kHz and 16kHz.

2.3.1 Advanced data

An another useful information available on the website is the speech segmentation. An accurate segmentation has been cut for each meeting using beams of each of the 8 elements of the microphone array.

Figure 2.4 shows the room configuration. P1, P2, P3, P4 represent the seats where each participant is sit down, WH is the area where are given the white-board presentations, and finally PR is the area where are the projector screen presentations.

The segmentation files are organized as follows: anytime a sound is found in one of the six area, the start time and duration (in seconds) are stored in the corresponding file.

Examples: file WH contains:

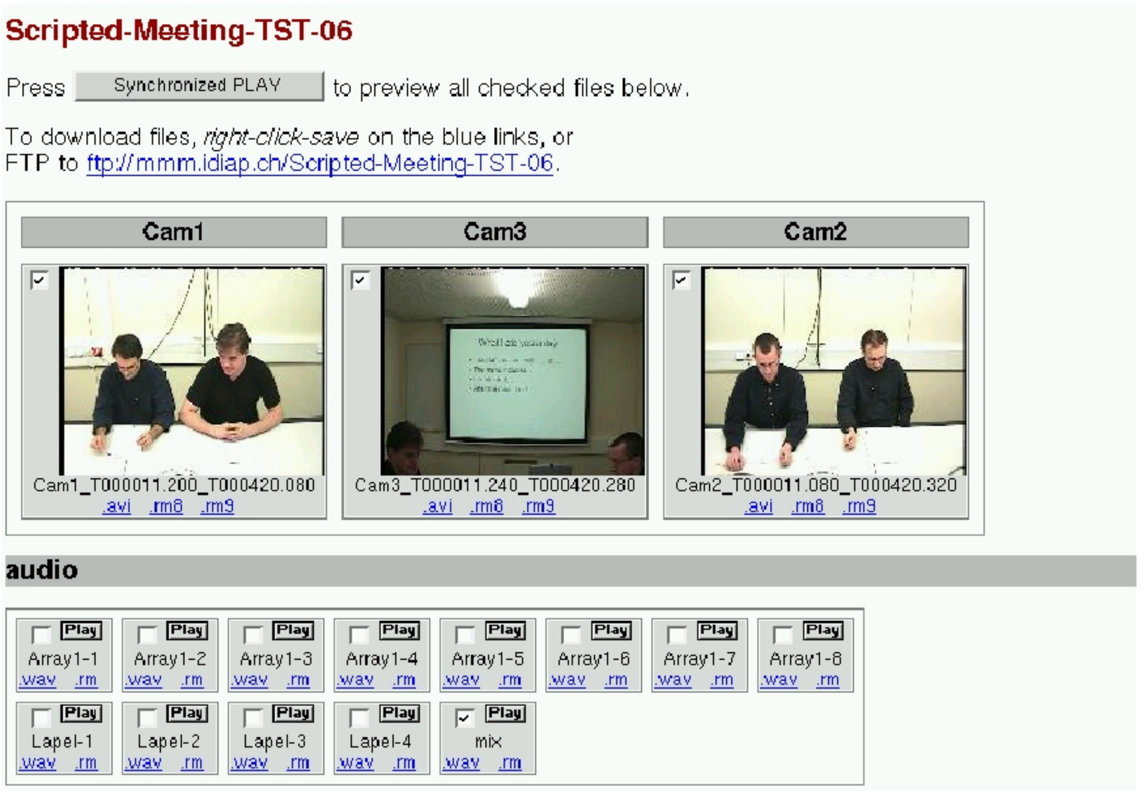


Figure 2.2: The mmm file server website

Figure 2.3: The three different cameras

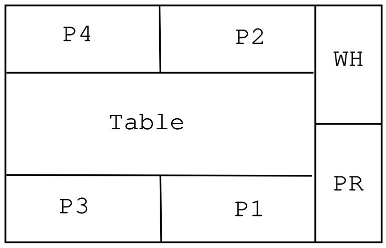


Figure 2.4: The configuration of the IDIAP Smart Meeting Room

111.112 00.160
111.848 00.144
112.712 00.224
113.080 00.160
113.912 00.176
...

i.e. at time 111.112 and for 00.160 seconds someone was doing a white-board presentation and so on. These data will be used later for a face-voice association in spite of the studied association whose details are given in appendix A.

Other information like the start time and duration of notes taking, discussions or monologues have also been saved for the purpose of research activities.

Chapter 3

Hierarchical and partitioning clustering techniques

3.1 What is clustering?

Clustering is a form of classification imposed over a finite set of objects [14]. The goal of clustering is to group sets of objects into classes such that similar objects are placed in the same cluster while dissimilar objects are in separate clusters. Representing data by fewer clusters necessarily loses certain fine details but achieves simplification. It represents many data objects by few clusters, and hence, it models data by its clusters. The problem of grouping [2] can be motivated by considering the set of points shown in the figures 3.1 and 3.2

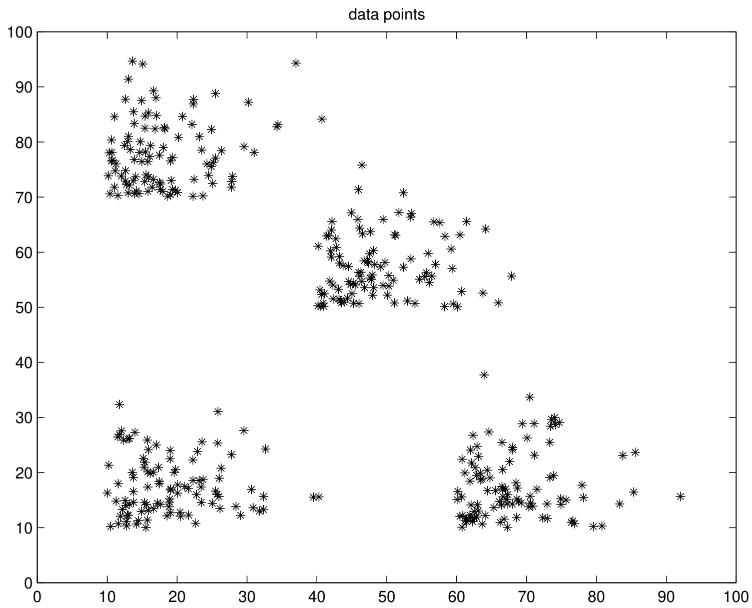


Figure 3.1: Example of different sets of data points

Typically a human observer will perceive 4 objects in the image 3.1. But how many in the figure 3.2? Four - a circular ring with a cloud of points inside it, and two clumps of points on its right.? Or

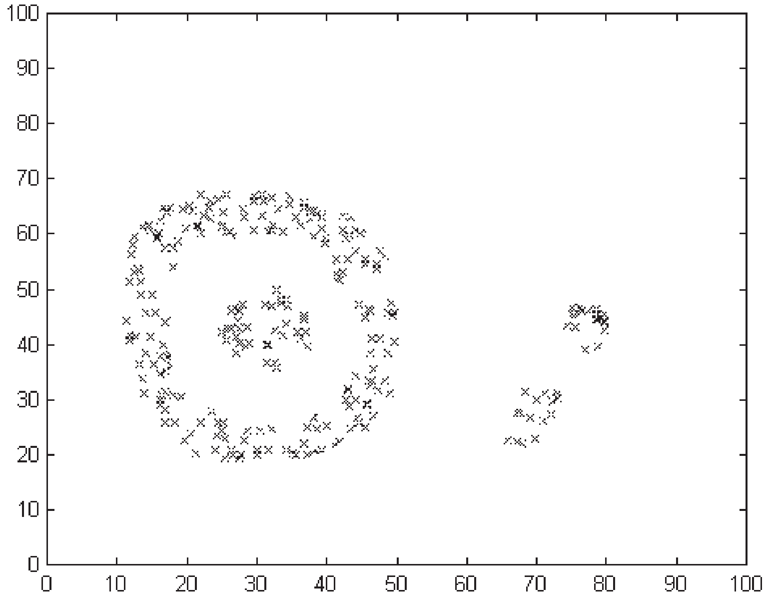


Figure 3.2: How many groups?

only three- two clumps on the right and a shaped object on the left? One even could argue that in fact every point is a distinct object. This problem shows that there are many possible partitions and that the similarity question must be well defined. Clustering (or classification) is a common form of data mining and has been applied in many fields such as statistics, pattern recognition, data compression, texture segmentation, vector quantization, computer vision and various business applications. Data mining [3] adds to clustering the complication of very large datasets with many attributes of different types. This imposes unique computational requirements on revelant clustering algorithms. Clustering algorithms can be classified into partitioning and hierarchical algorithms.

3.2 Hierarchical clustering

Hierarchical clustering builds a cluster hierarchy or, in other words, a tree of clusters, also known as *dendogram*. Hierarchical clustering methods are categorized into ***agglomerative*** (bottom-up)[6] (see figure 3.3) and ***divisive*** (top-down) [4](see figure 3.4). An *agglomerative* clustering starts with one-point (singleton) clusters and recursively merges two or more appropriate clusters. A *divise* clustering starts with one cluster of all data points and recursively splits the most appropriate cluster. The process continues until a stopping criterion (frequently, the requested number k of clusters) is achieved. Advantages of hierarchical clustering include:

- Exploration of data at different levels
- Ease of handling of any forms of similarity or distance

Disadvantages of hierarchical clustering are:

- Difficulty to define the termination criteria

- The fact that most hierarchical algorithms do not revisit once constructed (intermediate) clusters with the purpose of their improvement.

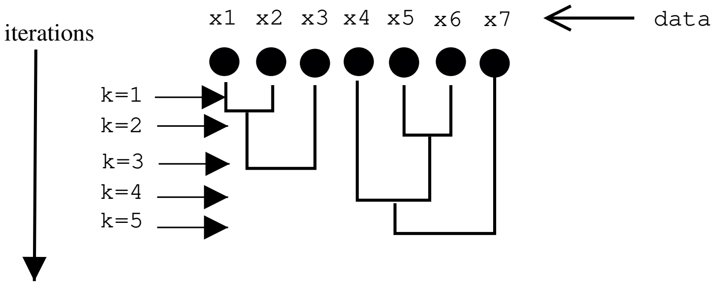


Figure 3.3: Evolution of agglomerative algorithm

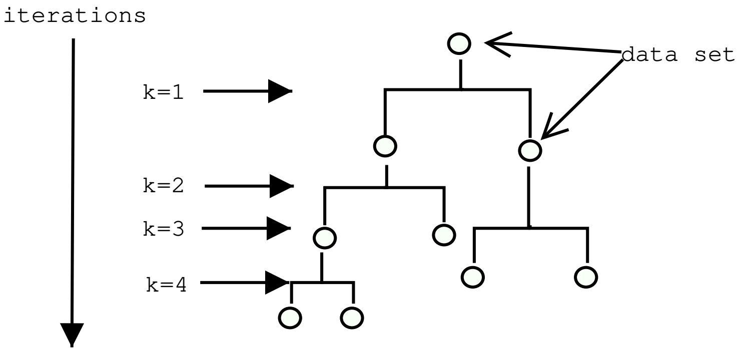


Figure 3.4: Evolution of divisive algorithm

Thus, both methods (agglomerative case and divisive case) proceed iteratively with merging or splitting of the most appropriate cluster(s) until the stopping criterion is achieved. The choice of a cluster(s) for merging/splitting depends on the (dis)similarity of cluster(s) elements. An important feature of (dis)similarity between two points is the distance between them. To merge or split subsets of points rather than individual points, the distance between individual points has to be generalized to distance between subsets. This new distance is called a ***linkage metric***. The type of the linkage metric used significantly affects hierarchical clustering. The (dis)similarity measure is comptuted for every pair of points with one point in the first set and another point in the second set:

$$d(\mathcal{C}_1, \mathcal{C}_2) = operation\{d(x, y) | x \in \mathcal{C}_1, y \in \mathcal{C}_2\}$$

where *operation* can be the minimum, maximum, average, median and so on. Linkage metrics-based hierarchical clustering suffers from time complexity. Despite this disadvantage, these algorithms are widely used.

3.3 Partitioning clustering

Partitioning algorithms create a partitioning of objects into a set of clusters. Because checking all possible subset systems is unfeasible, certain algorithm are used in the form of ***iterative optimization***. This means different ***relocation*** schemes that iteratively reassign points the k clusters. Unlike hierarchical methods, in which clusters are not revisited after being constructed, relocation algorithms gradually improve clusters. With appropriate data, this results in high quality clusters. As we already have seen (see section 3.2), pair-wise distances or similarities can be used to compute measures of inter- and intra-cluster relations. In iterative improvements, such pair-wise computations would be too expensive. Using unique cluster representatives resolves the problem. Depending on how these representatives are constructed, we can divide partitioning algorithms into ***k*-medoids** and ***k*-means** methods.

3.3.1 *K*-Medoids Methods

In k-medoids methods a cluster is represented by one of its points. Such representation has the following advantages: the choice of medoids is dictated by the location of the most important part of points inside a cluster and, therefore, it is less sensitive to the presence of outliers. When medoids are selected, clusters are defined as subsets of points close to respective medoids, and the average distance or another dissimilarity measure between its point and its medoid is used to find the closest medoid. Two early versions of k-medoid methods are the algorithm PAM (Partitioning Around Medoids)[4] and CLARA Clustering LARge Applications) [4]. PAM is iterative optimization that combines relocation of points between clusters with re-nominating the points as potential medoids. CLARA uses several (five) samples, each with 40+2 k points, which are each subjected to PAM. Further progress have been realized with the algorithm CLARANS (Clustering Large Applications based upon RANdomized Search) [5]. It is formalized as searching through a graph where each node is represented by a set of k medoids, and two of nodes are neighbours if they only differ by one medoid. The drawback of the k-medoids algorithms is the time complexity determining the medoids.

3.3.2 *K*-Means Methods

The ***k-means*** algorithm [7] is by far the most popular clustering tool used in scientific and industrial applications. The name comes from representing each of k cluster $\mathcal{C}_j$ by the mean (or weighted average) c_j of its points, the *centroid*. This can be negatively affected by a single outlier. On the other hand, centroids have the advantage of statistical meaning. The sum of differences between all the points of a subset and its centroid is usually used as intra-cluster distance similarity. L_2-norm can also be used: the sum of squares errors between the points and the corresponding centroids:

$$E(\mathcal{C}) = \sum_{j=1:k} \sum_{x_i \in \mathcal{C}_j} \|x_i - cj\|^2.$$

Two versions of k-means iterative optimization are known. The first one is similar to EM algorithm (for a quick introduction to EM algorithm, see [8]) and consists of two-step major iterations that (1) reassigns all the points to their nearest controids, and (2) recompute centroids of newly assembled groups. Iterations continue until a stopping criterion is achieved (for example, no reassigments happen). The second versions of k-means reassigns points based on effects moving a point from its current cluster to a potentially new one. If a move has a positive effect, the points is relocated and the two centroids are recomputed. The wide popularity of k-means algorithms is well deserved. It is simple and based on the analysis of variances. On the other hand, the k-means algorithms suffers from:

- the results strongly depend on the initial guess of centroids
- it is not obvious what is a good k to use

- the algorithm is sensitive to outliers

Of course it exists many other extensions and modifications and this method. It is not the goal of this report to see all of them.

Chapter 4

Audio Clustering

4.1 Speaker Clustering Algorithm

Our work on voice features extraction and audio clustering is based on a algorithm [11] developed at IDIAP by A. Jitendra. It is a HMM-based speaker clustering system, which includes a technique for automatically refining the number of clusters. Input parameters are *linear predictive cepstral coefficients* (LPCC) features built using *htk 3.1* software. Modifications have been provided to this method in order to fulfill our problem requirements.

4.1.1 Original Speaker Clustering Algorithm Overview

The clustering algorithm is based on an ergodic HMM with minimum duration constraints. Each state of the HMM represents a cluster and the *probability density function* (PDF) of each state (cluster) is represented by a *Gaussian mixture model* (GMM). The HMM is trained in an unsupervised manner using the *expectation maximization* (EM) algorithm. The initialization of the PDF's is done using the k-means algorithm.

It starts by over-clustering the data. The term "over-clustering" means that at the initial clustering step, data are deliberately clustered into a greater number of classes than the expected number of speakers in the data set. This reduces the probability that different speakers will be clustered into one class. The next step is to reduce the number of clusters by merging. At the end of the segmentation process (using Viterbi algorithm), the mutually closest pair of clusters is identified using a likelihood ratio distance measure, and these are then merged to form a single new cluster. This new cluster is then representing by another GMM and the parameters of this newly formed cluster are trained using the EM algorithm. In the next iteration, the segmentation is again found using the updated HMM with one less cluster and a new likelihood is computed. This likelihood increases if the two merged clusters are valid candidates for merging (the data in two clusters is from the same speaker). If, however, clusters having data from two different speakers are merged, the likelihood will decrease. The stopping criterion of the algorithm is the observation of this decrease.

4.1.2 Modifications

Before running the clustering method, a data pre-processing has been done. Starting from a single meeting audio signal (figure 4.1) and using segmentation files [1] (see section 2.3), we have computed an audio stream for each of the four participants in a given meeting. This has been done for each meeting. Thus 120 data streams (30 meetings and four persons in each meeting) are extracted and

[1] Remember that the segmentation files, based on a method of microphone-array beams analyzing, can define a reliable face-voice association: using them in relation with the position of the person is the image, can easily lead to a simple association between a speaker and a participant.

concatenated to form a single audio stream from which LPCC coefficients are computed. The segmentation knowledge of this stream into individual and homogeneous speaker streams is the great difference with respect to the method based on *Speaker Clustering Algorithm*. Thus the algorithm should be modified so that the segmentation boundaries do not change after the merging process. The main task is now to find the number of clusters (speakers) in the data.

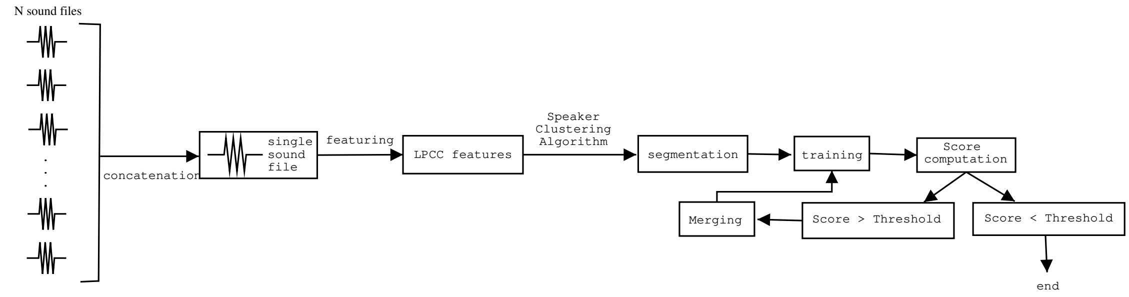


Figure 4.1: Speaker clustering algorithm overview

The algorithm starts by associating a cluster to each of the 120 segment of the input LPCC stream. GMM parameters are trained and the two closest clusters are identified (in the same manner as the original algorithm) and merged. But NO new segmentation is found. Only the data segments associated with the two merged clusters should be merged. When the stopping criterion of the original algorithm is achieved, each segment is assigned to a cluster (speaker). This means that ideally the final number of clusters should be the same as the number of participants involved in the 30 meetings.

Figure 4.2 shows an overview of the algorithm without segmentation change. The first line represents the audio file with N different segments based on a-priori knowledge that the four first segments are spoken by the four participants of the first meeting, the four following segments are spoken by the participants of the second meeting and so on. The second line depicts the assignment of each segment to a cluster. Then, merging is done until the stopping criterion is met with $n < N$ final clusters.

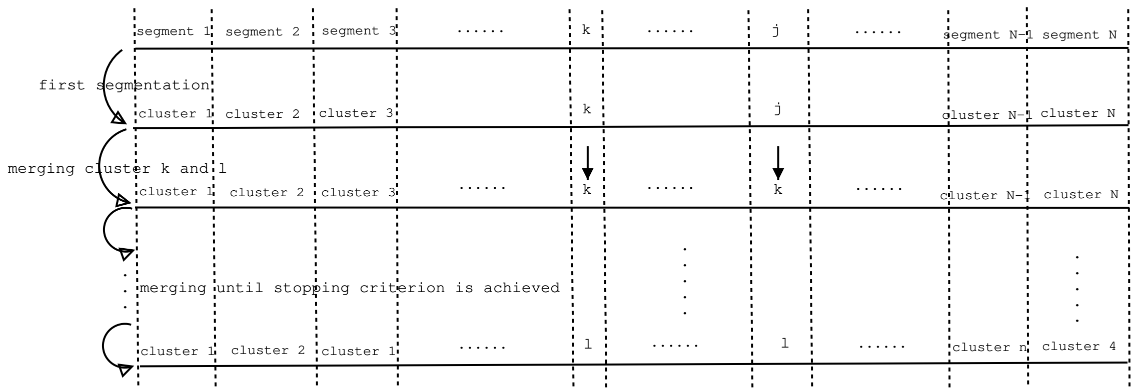


Figure 4.2: Speaker clustering algorithm without re-segmentation

4.1.3 Evaluation Criterion

We use the *average cluster purity* (acp) and the *average cluster purity* (asp) as explained in [11] to evaluate the results. First we define:

N_s: total number of speakers
N_c: total number of clusters
ns_i: number of segments spoken by speaker i
nc_i: number of segments in cluster i
ms_i: number of segments in cluster i spoken by the correct speaker
mc_i: number of segments in speaker i associated to the correct cluster

The correct speaker (respectively cluster) is defined as the one who (respectively which) appears the most in the segments in cluster (respectively speaker) i.

The average cluster purity *acp* is:

$$acp = \frac{1}{N_c} \sum_{i=0}^{N_c} ms_i / nc_i$$

Similarly, we calculate the average speaker purity *asp*:

$$asp = \frac{1}{N_s} \sum_{i=0}^{N_s} mc_i / ns_i$$

The *asp* gives a measure of how well a speaker is limited to only one cluster, and the *acp* gives a measure of how well a cluster is limited to only one speaker. In general, the *acp* decreases and the *asp* increases as the number of clusters decrease. In an ideal case (one cluster for each distinct speaker), both the *asp* and the *acp* are 1.

Finally the average evaluation criterion is: $K = \sqrt{acp * asp}$. The more closer to 1 K is, the better the results are.

4.1.4 Results

The iterative algorithm was initialized with 120 clusters and the number of GMM was set to 5. Seven different speakers are in the data set.

The next figures are obtained by iterating the modified algorithm until all segments are merged into a single cluster. The original stopping criterion stopped the merging with $n = 27$ clusters. Here we continued the process by removing the stopping criterion. Figure 4.3 shows the evolution of *asp* and *acp* as a function of the clusters number, and figure 4.4 shows the evolution of K. We can noticed that the greatest value of K is obtained with 8 clusters. When there are 7 clusters, which is the true number of speakers, the *asp* and *acp* values are still higher than 80%.

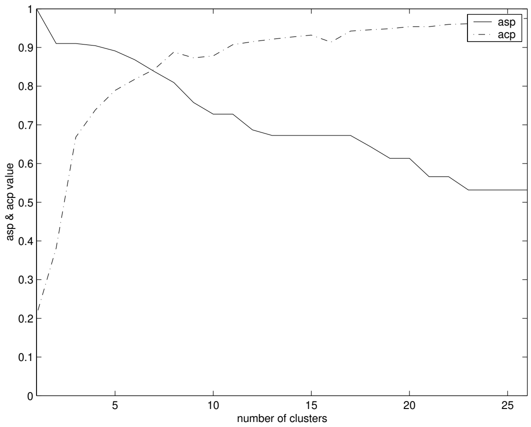


Figure 4.3: Evolution of asp and acp values

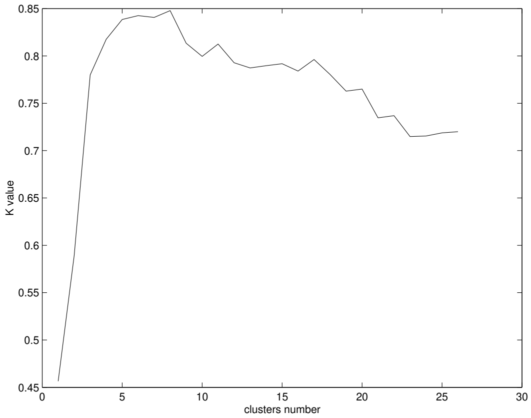


Figure 4.4: Evolution of $K = \sqrt{asp * acp}$

Chapter 5

Face similarity computation

5.1 Introduction

Face detection has several applications in areas such as content-based image retrieval, video coding, video conferencing, crowd surveillance. A general statement of the problem can be defined as follows: given an image, detect and localize an unknown number of faces. Two main approaches are used to solve this problem: feature-based approaches, which analyze low-level features, and image-based approaches, which deal with linear subspaces methods, neural networks or statistical approaches (read [15] for further information).

In our case, we compute face similarities in order to compare two faces and to define a distance between them. Two different situations are taken in account: comparisons between faces of a same person inside a same meeting and comparisons of different faces in different meetings. *Distance matrices* are built and provide the distances of all pairs of compared images. The definition of the distance criterion is also an important parameter and should be well chosen.

Two major problems appear in the definition of similarity between two faces: the first one is the illumination variation. This problem is less important when the two images are from the same meeting but can be important if it is not the case. Thus an image pre-processing is applied when we compare faces of a same person inside a same meeting. The second problem comes from the non-alignment of the two faces. For instance performing a pixel-to-pixel difference would not be optimal: even with two same images but the second one being translated with respect to the first one, the computed difference would return a great dissimilarity, which does not exist. I use a parametric displacement model estimation (a complete description is given in [9]), developed by Jean-Marc Odobez[1] during his PhD, to estimate displacement between the two faces and to compute a reliable face similarity distance. The last problem is hence solved because a registration is performed between both faces and greatly improves the similarity measures.

5.2 Image pre-processing

In order to reduce illumination variations which can occur between different meetings and even inside a same meeting with change of lightning intensities, images could be normalized ([10]) for illumination by applying a bandpass filter to the face regions, and scaling window's intensities to have mean zero and variance 1. In addition, faces are modified by multiplying them with a Gaussian centered at the center of the image.

-Bandpass: $B = I * G_{\sigma=1} - I * G_{\sigma=4}$

-Multiplication: $F(x,y) = B(x,y)e^{(-\frac{r(x,y)}{0.5})^2}$

Examples of this pre-processing are shown in figure 5.1.

[1] odobez@idiap.ch

a b

Figure 5.1: (a) original image, (b) pre-processed image

Pre-processed images are only used for clustering faces across meetings. When we cluster faces of a same meeting (cf chapter 6) in order to define reference images, gray-level images are used.

5.3 Description of robust-motion estimation algorithm

Image motion estimators are based on the estimation of a displacement vector in every point of an image. This algorithm cut the image in different areas in which motion can be described as 2D parametric models (affine or quadratic models). Then a multiresolution method is applied not on the original resolution image but on a pyramid of filtered and sampled images. The section 5.3.1 describes the 2D motion model and measure equation and the section 5.3.2 describes the least-square multiscale estimation method.

5.3.1 Motion model and measure equation

We focus on the 2D affine motion model. It can be defined by:

$$\left\{ \begin{array}{l} u_A(X_i) = a_1 + a_2 x_i + a_3 y_i \\ v_A(X_i) = a_4 + a_5 x_i + a_6 y_i \end{array} \right. \tag{5.1}$$

where $X_i = (x_i, y_i)$ is a pixel in the image and $\delta X_i = (u_A(X_i); v_A(X_i))$ is the displacement vector at point X_i parametrized by A.

This can be rewritten as:

$$\delta X_i = B(X_i)A$$

where

$$B(X_i) = \left[\begin{array}{cccccc} 1 & x_i & y_i & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_i & y_i \end{array} \right] \text{ and } A^t = (a_j)$$

This model can represent translation, rotation, zoom and even some deformation (shear). The algorithm assumes that the difference of intensity between a pixel in the first image and the corresponding displaced pixel in the second image is 0:

$$\frac{dI}{dt}(X_i, t) = 0$$

But as illumination variations can happen in face images we add overall intensity variation on the computed area:

$$\frac{dI}{dt}(X_i, t) = -\xi.$$

where ξ is a new parameter to estimate. Thus by introducing the variable r_i (and by removing time t for simplification):

$$r_i = \frac{dI}{dt}(X_i) + \xi \tag{5.2}$$

5.3.2 Least-square multiscale estimation

We first describe the least-square multiresolution estimation method based on Gauss-Newton algorithm which leads to an incremental estimation of motion model. In order to handle large motions, a coarse-to-fine method through each resolution level of the image pyramid is used.

Incremental Estimation

The equation 5.2 can also be deduced from the next relationship which is in fact the discretization of the equation 5.2.

$$r_i = I(X_i + \delta X_i, t + \delta t) - I(X_i, t) + \xi \delta t \tag{5.3}$$

where δX_i is the displacement, δt is a time interval between two consecutive images. We then choose by convenience $\delta t = 1$. We thus have to minimize the following function:

$$E_2(\Theta) = \sum_{X_i \in F} \rho(r_i)$$

where F is the block of estimated pixels, $\Theta^t = (A^t, \xi)$ and $\rho(r_i)$ can be defined by:

$$\rho(x) = \begin{cases} x^2 \\ \text{Beaton estimator} \\ \text{Cauchy estimator} \end{cases}$$

In order to minimize $E_2(\Theta)$ we assume that a current estimation $\widehat{\Theta}_k^t = (\widehat{A}_k^t, \widehat{\xi}_k^t)$ of Θ is known. We are now able to write:

$$\Theta = \widehat{\Theta}_k + \Delta\Theta_k$$

Therefore: $\delta X_i = B_i A = B_i \widehat{A}_k + B_i \Delta A_k$. Thus rewriting the new error function $E_2 = \sum_{X_i \in F} \rho(ri')$ where

$$ri' = I(X_i + B_i \widehat{A}_k + B_i \Delta A_k, t + 1) - I(X_i, t) + \xi$$

we can minimize E_2 with respect to $\Delta\Theta_k$ (see [9] for all details) and this leads to a refinement of $\widehat{\Delta\Theta}_k$

Coarse-to-fine Method

The last estimations are iterated through different levels of image resolution from the coarsest level to the finest. A L-level Gaussian pyramid is used, level 0 being the initial resolution level.

At coarsest level L, no estimation Θ is available, thus minimization of the next function is done:

$$E(\Theta) = \sum_{X_i \in F} \rho(r_i)$$

A refinement is then computed and when a stopping criterion based on the residual $\widehat{\Delta\Theta}^L$ is met (the increment is too small or a pre-defined number of iterations are done), parameters $\widehat{\Theta}^L$ are transmitted to the next lower level into the pyramid until $\widehat{\Theta}^0$ is achieved.

5.4 Distance definition

Many distances can be used in order to compute face similarities. We chose to focus on L_1- and L_2-norm:

$$d_1 = L_1 = \sum_{\widetilde{I}_2(x,y)} |I_1(x,y) - \widetilde{I}_2(x,y)|$$
$$d_2 = L_2 = \sum_{\widetilde{I}_2(x,y)} (I_1(x,y) - \widetilde{I}_2(x,y))^2$$
$$d_3 = \sum_{\widetilde{I}_2(x,y)} \min(|I_1(x,y) - \widetilde{I}_2(x,y)|, \alpha)$$

where $\widetilde{I}_2(x,y)$ are the interpolated pixels in registrated image, i.e. every pixel except the black-border ones (see figures 5.2). We use the distance d_3 in the further research topics. This distance is used in order to reduce influence of outliers, i.e. pixels which could not be re-positionned. We chose to set the value α to 30.

5.5 Application

The registration stage is very important in the visual clustering procedure because distance matrices (providing distances between all pairs of compared images) built are used later in face clustering, firstly between images of a same person in a given meeting and secondly across meetings. For instance, one of the main purpose of face clustering of a given person in a given meeting is to cluster faces with same heads poses and variations. If registration is well-performed, such images would have a small distance and would be more likely to appear in the same cluster.

Thus the estimation algorithm is used two times. Firstly for computing d_3 similarity measures between images of same face set (a face set includes several images a same person taken inside a same meeting). For that purpose, the images are not pre-processed: the algorithm works on gray-level images. Secondly for the computation of d_3 distance between images of different faces or different meetings. The images are at that time pre-processed. Finally the iterations are done with the following parameters:

- Initialization of translation parameters
- Weighted method: Beaton
- Final variance: 300
- Number of pyramid level: 4
- Starting level of constant parameters estimation: 4
- Starting level of linear parameters estimation: 1
- No quadratic parameters estimation
- Parameters estimated on a initial 60x74 rectangular-centred window (not on the full image).

Experiments show that using these level numbers for estimation reduce bad registration which could occur when the initial levels are used. Bad registration means that the algorithm is unable to find optimal parameters of a moved image with respect to one another. Beaton estimator used with a 300 final variance acts like a least-square estimation.

Level changing in the algorithm means that the estimation window resolution is increased by a factor of 2. In this case the estimation window size is 60x74 at level 0, 30x37 at level 1, ... , and finally 3x4 at level 4.

5.5.1 Example

The algorithm takes as parameters two images and return the displacement parameters A. Moreover a re-positionning of the second image is processed with respect to the first one. The algorithm computes the estimation model and applies it on the second image.

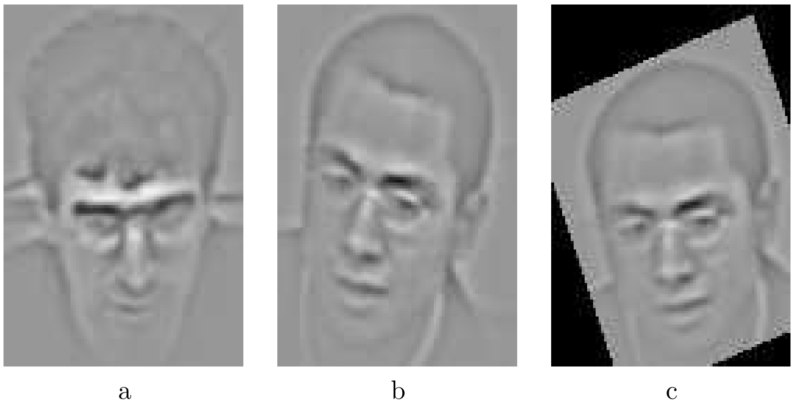


Figure 5.2: (a) (b) original images, (c) re-positionned image (b) after motion-estimation parameters application

Figure 5.2 shows in (a) and (b) the original images. Image (c) is the resulting image after applying on image (b) the motion-estimation parameters returned by the algorithm. The original image has been rotated and translated. We can see for example that the eyes (glasses) and the mouth of image (b) have been rotated in order to have the same orientation as in image (a).

5.5.2 Tests and results

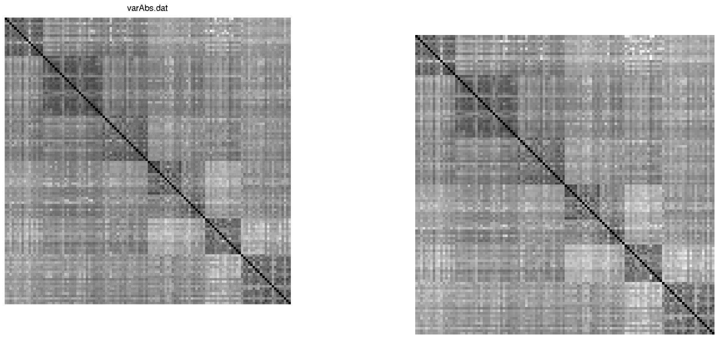
The first tests have been done on an independent database of 120 faces of seven people in order to analyze the algorithm behavior. Many parameters have been tested to improve the face similarities and clustering:

- Weighted method: Beaton or Cauchy
- Final variance
- Pyramid level where parameters estimation starts

Figures 5.3 and 5.4 shows results based on the independent database. Figures 5.3 shows distance similarities matrix using $L1$-norm and figure 5.4 shows the same matrix but using $L2$-norm. Pixel (i, j) represents the distance similarity between image i and j. A three-level pyramid has been used with the starting estimation levels set to 3 for the constant parameters and 2 for the linear parameters. The faces of a same person are gathered together by convenience. The blacker the pixels are, the closer images i and j are. Seven square blocks should appear along the diagonal, each block being faces of a same person.

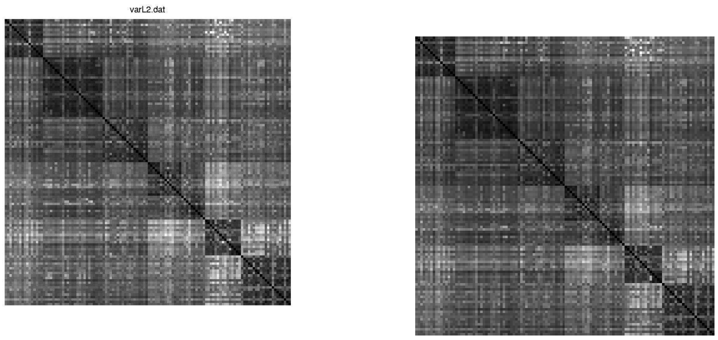
As experiments show that the translation parameters are the most important estimation so that registration is well-performed, we have noticed that initialization of these parameters could improve the translation registration. The following improvements are provided to the initial algorithm.

All translations, at positions inside a centered ellipse with radius of one quarter of the image size, are exhaustively tested. A pixel-to-pixel difference between the original image and the second translated image is calculated. We initialize the translation parameters with the displacement values whose difference is minimum. In order to reduce complexity, the horizontal and vertical motion steps are set to 2 and the difference is calculated on 2-down-sampling images.



a b

Figure 5.3: L1 distance matrix of 120 faces. (a) Weighted method: Beaton, variance= 250. (b) Weighted method: Cauchy, variance=40



a b

Figure 5.4: L2 distance matrix of 120 faces. (a) Weighted method: Beaton, variance= 250. (b) Weighted method: Cauchy, variance=40

Chapter 6

Face clustering

6.1 Introduction

Face clustering consists of three major steps: the first one builds face sets of each participant in each meeting, i.e. 30x4=120 sets with many face images as possible in each set. Second step builds face references for each set and finally last step clusters all reference faces. In first and last step, clustering is based on distances between faces. Computation of these distances is done by the algorithm of registration (see chapter 5) on gray-level images for the first clustering and on pre-processed images (as explained in section 5.2) for the second clustering.

6.2 First step- Face sets extraction

As the main task in video person clustering is to find faces *reference* images that show different head poses so that it improves further comparisons between different persons, the first step in order to find these images is to extract as many faces as possible from video files. The first step builds a face set for each participant of each meeting.

Figure 6.1 shows an overview of the face extraction procedure.

First (a-b) images are extracted from a server video files at a rate of 1/5, i.e. one frame out of five. Each meeting consists on 2 video files, one for the camera pointing on the left-side table and the other for the camera pointing on the right-side table. A frame extraction software developed at IDIAP (vitcmovie2ppm) is used and the images are saved with ppm extension. The images are then cropped (c) in order to reduce the face detection algorithm complexity. The borders of the cropped images are always the same because the chairs and persons positions never change. A face detection algorithm also developed at IDIAP by S. Marcel is applied on each image (d). Ten patterns of detection are returned by the algorithm, each one with the upper-left corner position and a reliability score. Three processing are sequentially applied on all patterns of **all** images of one camera meeting. The first processing removes patterns whose score is not high enough. A flexible threshold is set in order to have a sufficient number of faces because scores highly depend on head types (bearded man, a lot of hair,...) -the algorithm is based on a skin detection. The second processing divides the patterns in two groups based on their positions, those who are in the left part of the images and those who are in the right part (e). Each group is then treated separately. Finally the last processing is applied on the patterns in order to save only faces “clients” and not false detections like arms or wall. Each pattern group is clustered using *k-means* algorithm with $k = 2$ applied on the pattern centers and the patterns kept are the ones whose distance between their y-position center and the y-position center of the **upper** cluster is less than 90 pixels (g). This number 90 has been experimentally set.

Finally (h) the face images are extracted from each client pattern.

Each participant in each meeting has thus a face image set.

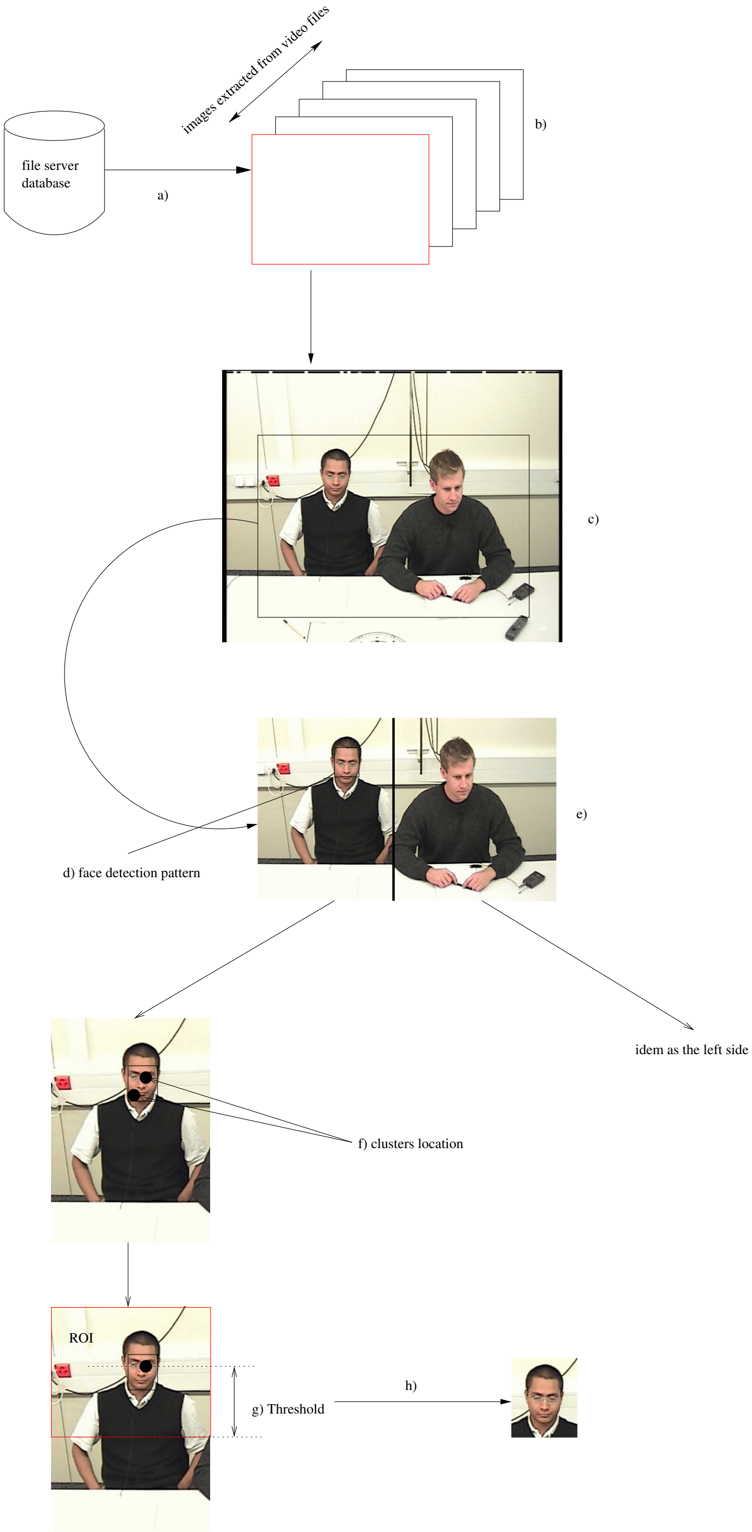


Figure 6.1: First step algorithm overview

6.3 Second step- Face references computation

The search of the number of participants, and person clustering are based on a face *reference* images clustering. Given a **distance similarity matrix** $\mathcal{M}$ (corresponding to the distance between all faces of a same set), the goal is to find **reference** images based on the matrix $\mathcal{M}$. The references should represent a complete set of different head poses. Further comparisons between different sets are indeed based on the reference images. Comparisons between two persons (or two face sets) are indeed improved if the compared faces have the same orientation and the likelihood of founding two (or more) such faces is improved if the reference sets have many variations. This is the reason of trying to find two references with right-turned heads, two with left-turned heads and one with a centered head.

The matrix $\mathcal{M}$ is computed using the motion estimation algorithm (see chapter 5). $\mathcal{M}(i, j)$ is the distance between images i and j, both images belonging to the same face set.

Here is a complete description of the procedure of reference images definition. First an agglomerative hierarchical algorithm is applied on matrix $\mathcal{M}$. The major steps in agglomerative clustering are contained in the following procedure, where c is the desired final number of clusters:

- (Agglomerative Hierarchical Clustering [12])

$$\textbf{begin} \quad \textbf{initialize} \quad c, \widehat{c} \leftarrow n, \mathcal{D}_i \leftarrow x_i, i = 1...., n \tag{6.1}$$
$$\textbf{do} \quad \widehat{c} \leftarrow \widehat{c} - 1 \tag{6.2}$$
$$\text{find nearest clusters, say } \mathcal{D}_i \text{ and } \mathcal{D}_j \tag{6.3}$$
$$\text{merge } \mathcal{D}_i \text{ and } \mathcal{D}_j \tag{6.4}$$
$$\textbf{until} \quad c = \widehat{c} \tag{6.5}$$
$$\textbf{return} \quad c \text{ clusters} \tag{6.6}$$
$$\textbf{end} \tag{6.7}$$

Four different measures are tested to compute the distance between any two clusters, and hence to find the "nearest" clusters required in the algorithm. These measures are the following:

$$d_{min}(\mathcal{D}_i, \mathcal{D}_j) = \min_{\substack{x \in \mathcal{D}_i \\ x' \in \mathcal{D}_j}} (\|x - x'\|)$$
$$d_{med}(\mathcal{D}_i, \mathcal{D}_j) = \operatorname{median}_{\substack{x \in \mathcal{D}_i \\ x' \in \mathcal{D}_j}} (\|x - x'\|)$$
$$d_{max}(\mathcal{D}_i, \mathcal{D}_j) = \max_{\substack{x \in \mathcal{D}_i \\ x' \in \mathcal{D}_j}} (\|x - x'\|)$$
$$d_{mean}(\mathcal{D}_i, \mathcal{D}_j) = \frac{1}{n_i n_j} \sum_{x \in \mathcal{D}_i} \sum_{x' \in \mathcal{D}_j} (\|x - x'\|)$$

The "nearest clusters" means that the two clusters whose distance $d_{.}(\mathcal{D}_i, \mathcal{D}_j)$ is minimum are merged. All distances have been tested but the experiments show that d_{mean} gives slightly better results. As described, this procedure terminates when the specified number of clusters has been obtained. We set this number to 20.

During the second step the five most numerically-important clusters are kept and medoids from each of the five clusters are computed as follows: the medoids of a cluster is the element whose distance from itself to the most farthest element is minimum.

$$d = \min_{x \in \mathcal{D}_i} (\max_{x' \in \mathcal{D}_i} (\|x - x'\|)) \qquad (6.8)$$

It can also be seen as follows: given one element, let us define the circle including all points whose radius is minimum and whose center is the element itself. The medoid of a cluster is the element whose circle radius is the smallest (see figure6.2).

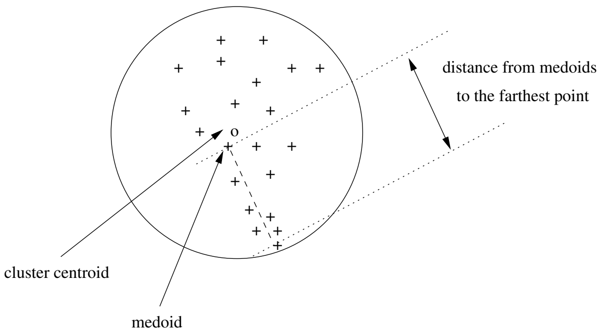


Figure 6.2: Illustration of the cluster medoid search

Refinements are provided to the first clustering by applying a k-medoids on the initial matrix using the five medoids found in the previous step as initial clusters. Here is the procedure of Partitioning Around Medoids (PAM) (in this case, *object* has the same meaning as *image*):

- (Partitioning Around Medoids [13])

begin	**initialize**	choose k medoids from T objects randomly
	Evaluation	calculate the cost $D'_t - D_t$ for each swap of one medoid with one object, where D_t is the total distance before swap and D'_t is the total distance after swap
	Selection	accept the swap with the best cost and if the cost is negative, go again to Evaluation step; otherwise record the medoids and terminate the program
end		

The algorithm has been slightly modified: the medoids in the initialization step are not randomly selected but are the medoids found at the end of the hierarchical algorithm. The cost between two objects (images) is just the distance between them (given by the corresponding value in the matrix $\mathcal{M}$).

Thus the algorithm returns five clusters in which medoids are found in the same manner as above: it is the element whose distance from itself to the farthest element is minimum. The five images found are set as the reference images.

6.3.1 Results

This section shows results of the implemented different clustering algorithms. Figures 6.3 to 6.6 show the obtained clusters and medoids (of one face set) with the clustering algorithms. Figure 6.3 shows the five greater clusters of the hierarchical clustering. Figures 6.5 displays the medoids of each cluster. Figures 6.4 shows the five clusters after PAM algorithm and figure 6.6 displays the resulting reference images. We can notice in this example the improvements provided to the medoids search: the set of medoids after the PAM algorithm is more varied and heterogeneous than the medoids after the hierarchical clustering. We can also notice that heads poses inside clusters (after PAM) are almost unvaried.

Figure 6.7 displays examples of reference images obtained by the above method. Many set reference images are what we expected, i.e. a set of different head poses characterizing the appearances in the meeting. Unfortunately some images do not fulfill the expectations: similar head poses between a same set can be found or worst arm images are selected as medoids! This means that participant head has sometimes few variations all along a meeting and not enough different poses can be detected. Problems can also occur when not enough detections are found by the skin-based face detection algorithm. In these cases, the sets hold sometimes less than 50 images and clustering on a such few data gives unstable results. In addition, the face detection algorithm is optimal when images contain frontal faces and it is not necessary the case in our database. It is moreover highly dependent of the skin color. When not enough skin-color pixels are available, typically when somebody has long hair on the forehead or is bearded, the face is not selected. This explain why only images with a hand near the face are selected for a man with a beard. In addition and despite all processing for removing arms detections, 7 arms are still selected as medoids (7 out of 600 images) due to the small distance between arm and face.

Figure 6.8 shows three examples of medoids registered with respect to one medoid using the registration algorithm (read chapter 5). First line displays original medoids and second line shows re-positionned images with respect to the images on the right. Sometimes the estimation parameters are completely false (third image of the first example and second image of the second example). This can easily be explained by the head pose in first case (frontal head) and by the fact that the algorithm tries to match an arm on a face in the second case.

6.4 Third step- Face clustering across meetings

In this stage all distances between all reference images of all sets are computed and stored in a distance matrix (as explained in chapter 5). Comparisons are done on pre-processed images. As we have 120 sets of five reference images, 600 x 600 distances have been calculated. The last step of face clustering is based on the assumptions that same faces would have great similarity distances and dissimilar faces would have great dissimilarities. Two hypothesis are set in order to achieve the face clustering task:

Hypothesis H0: face sets belong to the same person

Hypothesis H1: face sets do not belong to the same person

Two models of distance-based probabilities are trained:

$$p(d(S_i, S_j)|H0) \text{ and } p(d(S_i, S_j)|H1)$$

The goal of this section is to compute a similarity score $\frac{p(d(S_i,S_j)|H0)}{p(d(S_i,S_j)|H1)}$ and to take a threshold-based decision whether set S_i and S_j are the same person or are not.

The first problem is the definition of $d(S_i, S_j|H0)$ and $d(S_i, S_j|H1)$. Ten meetings have been randomly selected as train sets. All possible combinations of sets matching hypothesis H0 are used for training $d(S_i, S_j|H0)$ and $d(S_i, S_j|H1)$ is trained in the same manner. I visually defined whether two sets are a same person or are not. Then 2 different distances between train sets have been defined. The first one takes the 3 smallest values as distances between two sets and the second one takes only the smallest distance. Next sections describe both methods and results obtained on the 30 meetings used as test sets.

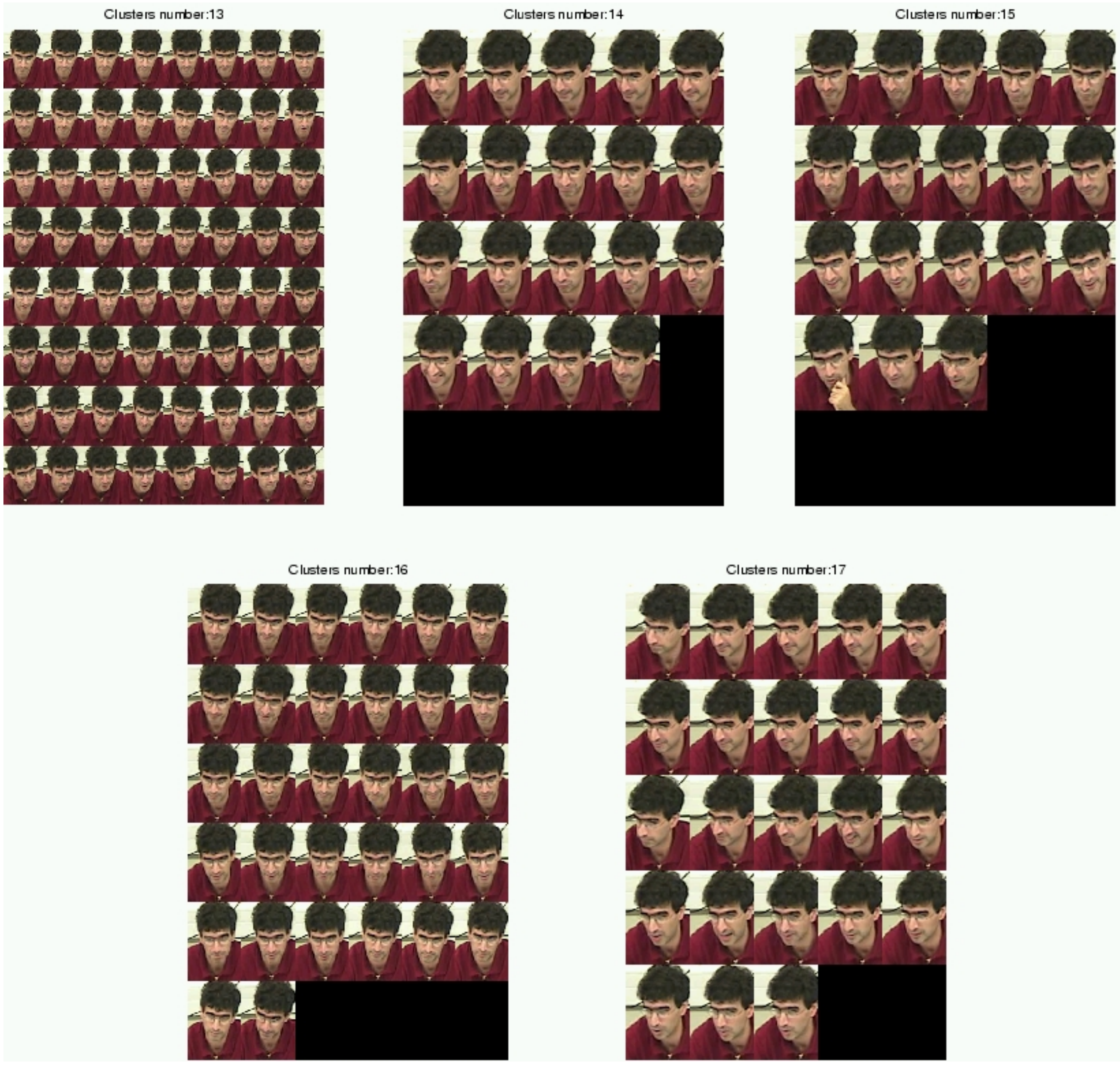


Figure 6.3: The five greater clusters after hierarchical clustering (as a consequence, all the images of the dataset are not displayed)

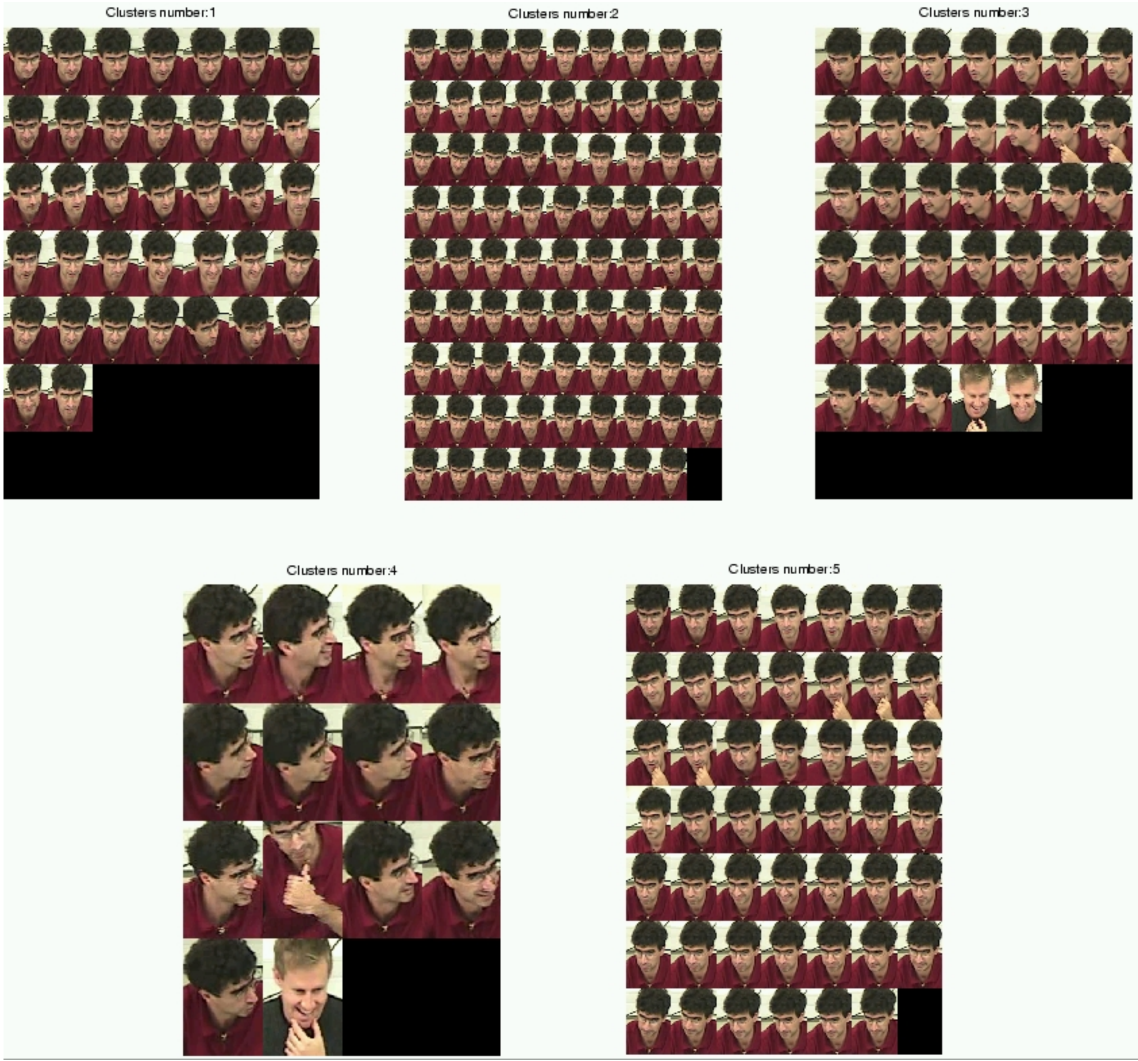


Figure 6.4: All clusters after PAM algorithm

Images medoids after hierarchical clustering

Figure 6.5: The five medoids resulting in hierarchical clustering

Images medoids after PAM

Figure 6.6: Reference images

Figure 6.7: Examples of reference images

Figure 6.8: Examples of registration with respect to one medoid

6.4.1 The three minimum distances method

In the first case we define distances between two sets as the 3 smallest distances between each images pair of both sets (remember that there are 5 images per set).

$$d(S_i, S_j) = \{d_1, d_2, d_3\} \tag{6.9}$$

where

$$\begin{aligned} d_1 &= \min_{x \in S_i, y \in S_j} \|x - y\| \\ d_2 &= \min_{x \in S_i, y \in S_j} (\|x - y\| \backslash \{d_1\}) \\ d_3 &= \min_{x \in S_i, y \in S_j} (\|x - y\| \backslash \{d_1, d_2\}) \end{aligned}$$

Once the distances have been computed, they are modelized by a Gaussian Mixture Model (GMM) using *Torch*[1], a software developed at IDIAP, and the probabilities $p(d(S_i, S_j)|H0)$ and $p(d(S_i, S_j)|H1)$ are given by the modelized GMM:

$$p(d) = \sum_i w_i \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp - \frac{(x - m_i)^2}{2\sigma_i^2}$$

Figure 6.9 shows an 15-bin histogram of the distances matching H0 (respectively H1) and the modelized functions $p(d)$ with 5 Gaussian.

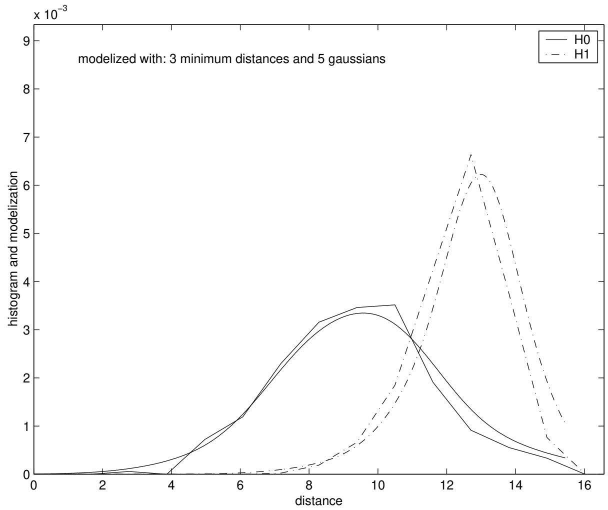


Figure 6.9: Histogram and modelization of set distances using the 3 smallest values

Let us now define the ratio $\frac{p(d(C_i,C_j)|H0)}{p(d(C_i,C_j)|H1)}$ between 2 clusters :

$$r(C_i, C_j) = \frac{1}{nb\_min} \frac{\prod_{k=1}^{nb\_min} p(d_k(C_i, C_j)|H0) * p(d_k(C_j, C_i)|H0)}{\prod_{k=1}^{nb\_min} p(d_k(C_i, C_j)|H1) * p(d_k(C_j, C_i)|H1)} \tag{6.10}$$

[1]www.torch.ch

where $d(C_i, C_j)$ is defined similarly as in equation 6.9, and let us now take the logarithm:

$$\log(r) = \frac{1}{nb\_min} \sum_{k=1}^{nb\_min} [\log(p(d_k(C_i, C_j)|H0)) + \log(p(d_k(C_j, C_i)|H0))] - \frac{1}{nb\_min} \sum_{k=1}^{nb\_min} [\log(p(d_k(C_i, C_j)|H1)) + \log(p(d_k(C_j, C_i)|H1))] \quad (6.11)$$

where nb_min is the number of minimum distances kept (3 in this case). $p(d_k(S_i, S_j)|H0)$ and $p(d_k(S_i, S_j)|H1)$ are given by the values of the corresponding modelized functions. Please note that C_i means the cluster i and S_i means the set i. The set i includes all the reference images of a given person in a given meeting. Many sets can be included in cluster i.

Figure 6.10 shows a 15-bin histogram of the ratio defined above between all pairs of train sets matching hypothesis H0 (full-line) and all pairs of train sets matching hypothesis H1 (dotted lines). Figure 6.11 shows the cumulative error with respect to the ratio used as threshold. Error is defined as the sum of pairs matching H0 and whose distance is smaller than a given threshold and pairs matching H1 and whose distance is greater than the same threshold.

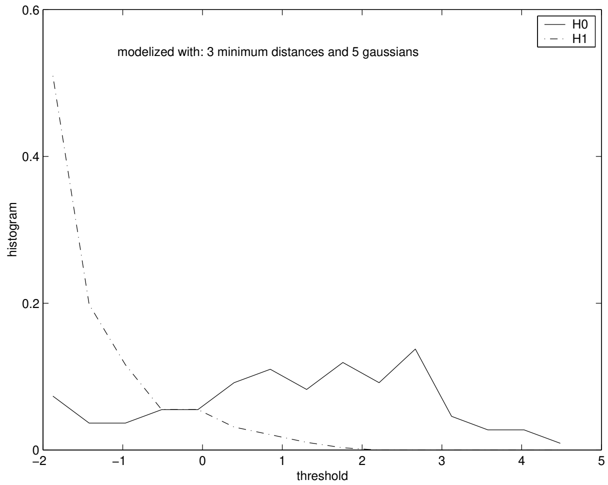


Figure 6.10: Histogram of $\log(r)$ for sets pairs matching H0 ('-') and sets pairs matching H1('-.')

Finally the results are tested on all the 30 meetings (including the meetings used to train the parameters). A hierarchical clustering (see chapter 6 and section 3.3) is applied on the 120 sets (all sets of all meetings). However, in this case, the two closest (and merged) clusters are the ones whose log ratio is the highest and the distances used for computing the logarithm ratio (given by equation 6.11) are defined as the 3 smallest values (equation 6.9).

Evaluation criterion

The evaluation criterion is defined similarly to section 4.1.3 and measures how well a cluster is limited to one participant and how well a person is limited to one cluster.

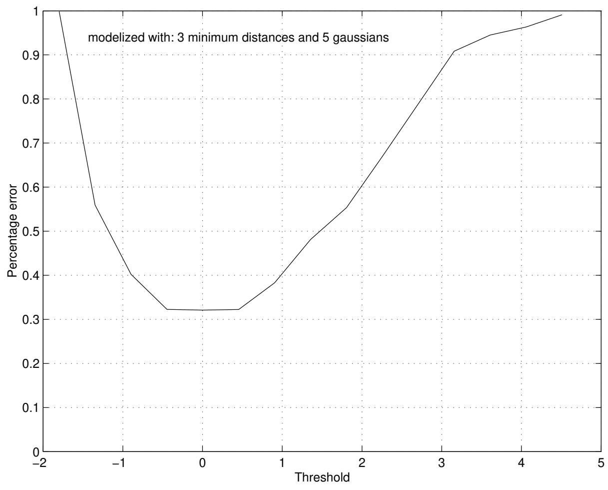


Figure 6.11: Cumulative error with respect to the ratio

N_p: total number of participants
N_c: total number of clusters
N_s: total number of sets (=120)
np_i: number of sets of participant i
nc_i: number of sets in cluster i
mp_i: number of sets in cluster i associated to the correct participant
mc_i: number of sets in participant i appearing in the correct cluster

The correct participant is the set appearing the most in cluster i and vice-versa.

The average cluster purity acp is:

$$acp = \sum_{i=0}^{N_c} w_i * \frac{ms_i}{nc_i}$$

where w_i is defined by: $w_i = \frac{nc_i}{N_s}$. We can rewrite:

$$acp = \frac{1}{N_s} \sum_{i=0}^{N_c} ms_i$$

Similarly, we calculate the average speaker(participant in this case) purity asp:

$$asp = \frac{1}{N_p} \sum_{i=0}^{N_s} mc_i/np_i$$

Figure 6.12 shows the evolution of asp and acp. Figure 6.13 shows the new results obtained by the addition of the following constraint in the hierarchical clustering: two clusters can not be merged if one set of the first cluster and one set of the second belong to the same meeting. In this special case, the algorithm stopped at 7 clusters and clustering is perfect: both asp and acp values are 1.

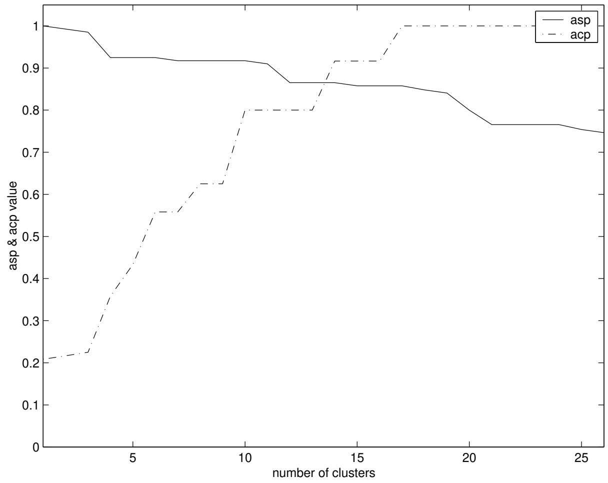


Figure 6.12: Evolution of asp and acp with respect to the final number of clusters

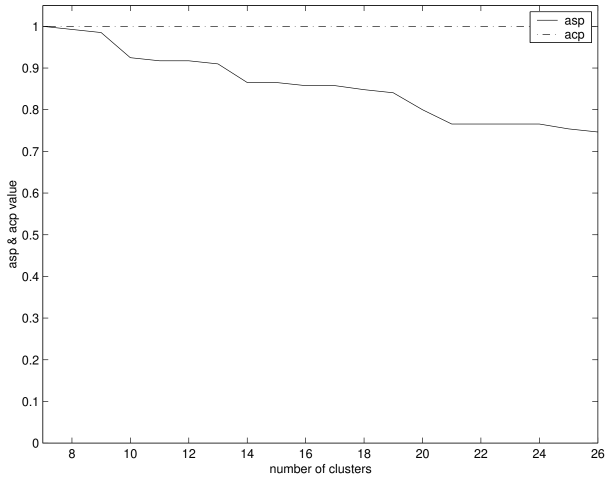


Figure 6.13: Evolution of asp and acp with the addition of a new constraint in the hierarchical clustering

6.4.2 The minimum distance method

This method differs from the first one with respect to the distance measure calculation. When GMM are trained on the 10 train meetings, only the smallest distance between two sets is kept.

$$d(S_i, S_j) = \min_{x \in S_i, y \in S_j} \|x - y\| \tag{6.12}$$

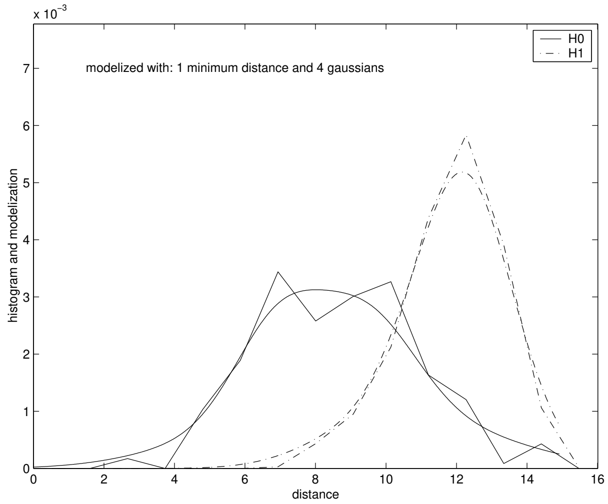


Figure 6.14: Histogram and modelization of sets distances using the smallest value

Figure 6.14 shows the histogram and the GMM modelizations based on these new distances. $p(d(S_i, S_j)|H0)$ and $p(d(S_i, S_j)|H1)$ are defined in the same manner as in the first method and figures 6.15 and 6.16 show the new logarithm ratio histogram and cumulative error.

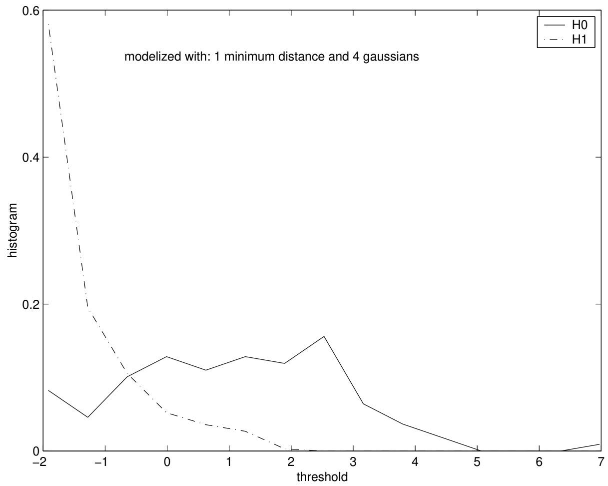


Figure 6.15: Histogram of $\log(r)$ for sets pairs matching H0 ('-') and sets pairs matching H1('-.')

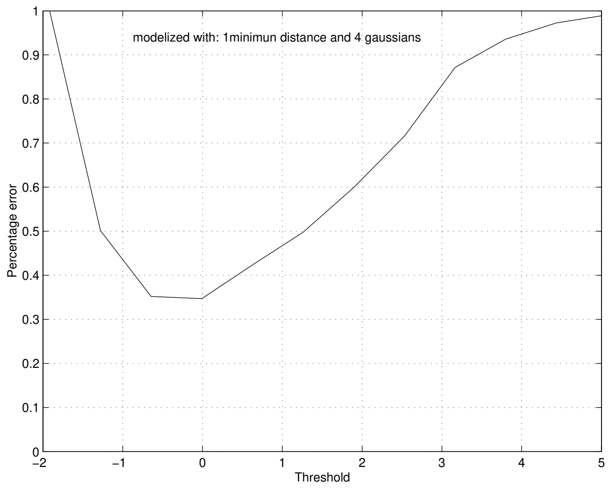


Figure 6.16: Cumulative error with respect to a given threshold

Finally the results are also tested on the 30 meetings and a hierarchical clustering based on logarithm probabilities is applied. But the main difference with respect to the first case is about the logarithm probability computation between two clusters:

$$r(C_i, C_j) = \frac{1}{2 * \mathrm{card}\{C_k\} * \mathrm{card}\{C_l\}} \frac{\prod_{S_k \in C_i} \prod_{S_l \in C_j} p(d(S_k, S_l)|H0) * p(d(S_l, S_k)|H0)}{\prod_{S_k \in C_i} \prod_{S_l \in C_j} p(d(S_k, S_l)|H1) * p(d(S_l, S_k)|H1)}$$

And taking logarithm:

$$\begin{aligned} log(r(C_i, C_j)) \quad = \quad & \frac{1}{2 * \mathrm{card}\{C_k\} * \mathrm{card}\{C_l\}} \sum_{S_k \in C_i} \sum_{S_l \in C_j} [log(p(d(S_k, S_l)|H0)) + log(p(d(S_l, S_k)|H0))] \\ & - \sum_{S_k \in C_i} \sum_{S_l \in C_j} [log(p(d(S_k, S_l)|H1)) + log(p(d(S_l, S_k)|H1))] \end{aligned} \qquad (6.13)$$

where distance $d(S_i, S_j)$ is the smallest value among all distances of every image pair of sets i and j (equation 6.12).

Figures 6.17 shows the evolution of the asp and acp resulting in method 2.

Finally figure 6.18 shows evolution of K defined as $K = \sqrt{asp * acp}$ for the both studied distance measures. We can notice that results of K values are better from a final number of clusters of 16.

Figures 6.19 shows the logarithm ratio value when 2 clusters are merged for both methods. The sign of the threshold can be seen as a stopping criterion (see figure 6.19): indeed, with such a criterion, method 2 would stop at 8 clusters. Remember that the true number of participants is 7. Using the first method and the same stopping criterion, the algorithm would only stopped at 1.

The comparisons between the asp and acp values show that the second method is better. Indeed, the acp is still 100% when there are 11 clusters. With the first method, the acp drops from 16 clusters. Let me now analyse the results of the first method when there are 17 clusters and acp is still 100%: 6 out 7 participants are well associated, i.e. the asp and acp for these 6 people are both 100%, but the last person is clustered in 11 groups and thus the asp drops because of this person. Reader must know that we had many problems by extracting enough reliable faces of the same “not-well clustered” person mainly because face detector had difficulties founding his face (it is a man with a beard and long hair on the forehead). The reference images of this person are thus bad and often include a hand near his face. The registration algorithm can not return reliable results in these conditions and the

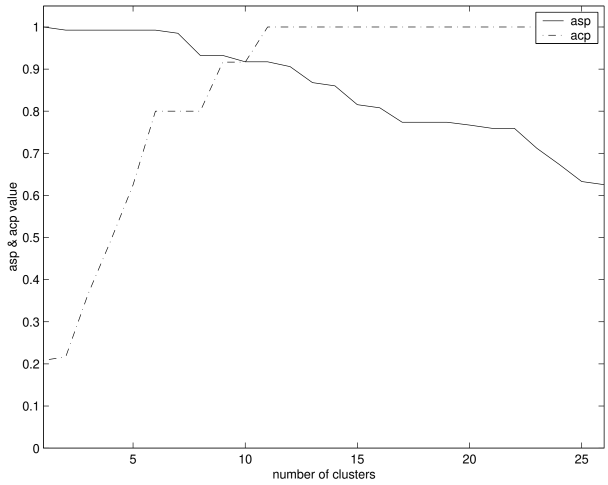


Figure 6.17: Evolution of asp and acp with distances between sets defined by the minimum value

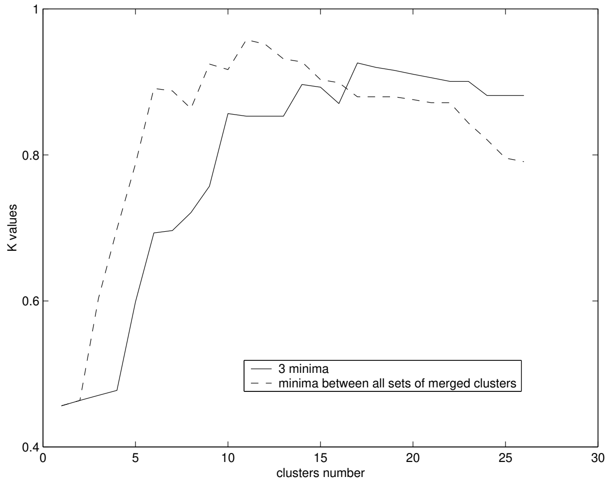


Figure 6.18: Evolution of K with both distance computation method

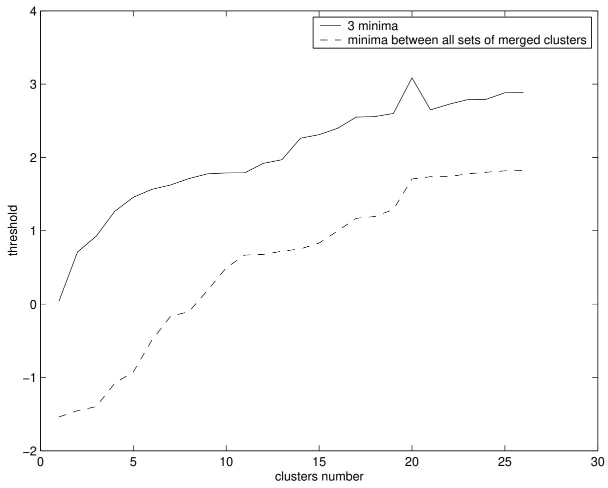


Figure 6.19: Evolution of logarithm ratio when 2 clusters are merged

similarity calculations give great distances between images and sets of this participant. Therefore face clustering could be almost perfect if the faces extractions could be improved.

We could finally say that the results are distorted because the test sets include all the train sets. In order to prove the real efficiency of the algorithm, a second class of tests is done: the GMM parameters are trained in the same manner as in the second method but testing is done on the 20 remaining meetings (independently of the 10 training meetings). Figure 6.20 shows the evolution of the asp and acp for the new testing sets. We can notice that the evolution is similar to the one based on the 30 meetings.

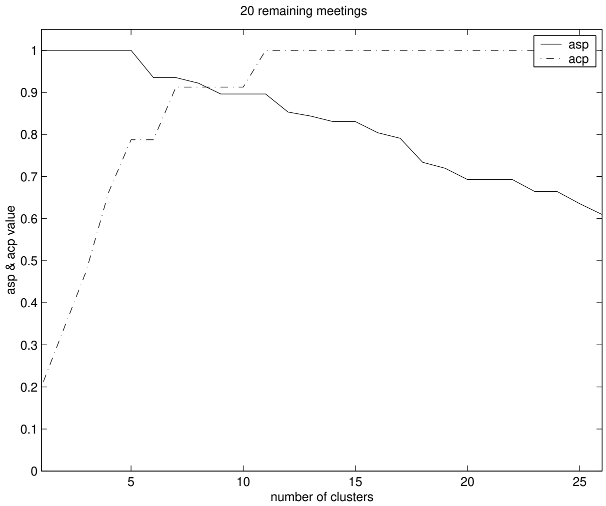


Figure 6.20: Evolution of the asp and acp based on independent test sets with respect to the train sets

Chapter 7

Merge

7.1 Introduction

Neither audio clustering nor face clustering gives absolutely reliable results. We can however notice that face clustering gives good results, but on the other hand, many constraints are added during the three steps of clustering like the assumptions that the person motions are small and that their positions are almost unvarying. Audio clustering has thus less constraints and is more reliable on the long time.

Person clustering is based on the fact that a person is characterized by a voice and a face. It hence combines audio and face data in order to improve results. The *speaker clustering algorithm* has been once again modified in order to insert the second method of face clustering during the likelihood ratio evaluation process.

7.2 *Speaker clustering algorithm* modifications

The original *speaker clustering algorithm* computes a likelihood ratio for the 2 candidate segments for merging. It is defined as the difference between likelihood before and after the merge of candidate segments. Finally the two merged clusters are the ones whose likelihood difference is the highest.

I modified the algorithm in order to include in this process the face information and similarity resulting from the second method of face clustering (see 6.4.2). The audio log-likelihood ratios have first to be normalized in order have same order of magnitudes than the face-based ones. The audio likelihoods are normally depending on the number of frames in the audio data segments and their ratios are hence divided by the number of frames in the two compared audio segments. Let us now define by $S_a(i,j)$ the audio likelihood ratio of the candidate clusters i and j, and by $S_v(i,j)$ the logarithm ratio (using second method) of faces of the participants corresponding to the audio clusters i and j. The new multimodal log likelihood ratio is given by:

$$S_{new}(i,j) = \beta * \alpha_a(\frac{S_a(i,j)}{nbf}) + (1-\beta) * \alpha_v S_v(i,j)$$

where nbf is the sum of the frame numbers in segment i and j, β is a weighted factor, α_a and α_v are magnitude coefficients. α_a and α_v are chosen so that the magnitudes of $\alpha_a * (\frac{S_a(i,j)}{nbf})$ and $\alpha_v S_v(i,j)$ are similar. We experimentally set these coefficients to 5/2 for α_a and 1/10 for α_v. Finally experiments are done with $\beta = 0.8$ and $\beta = 0.5$.

7.3 Results

Figure 7.1 shows the evolution of asp and acp from 5 clusters to 30 by combining the face and audio data in the likelihood ratio evaluation. Asp and acp are defined similarly as in section 6.4.1. The red curve shows the evolution when β is set to 0.5 and the blue curve is when β is set to 0.8. We can notice the improvements provided with respect to the audio clustering results (see section 4.1.4), whose results are really better now, but also with the face-based clustering results (section 6.4.2): the value of acp drops below 1 from 10 clusters for the audio-video-combination-based clustering ($\beta = 0.8$), and the same criterion drops below 1 from 11 clusters for the face clustering results. Remember that face-only clustering performs a correct association for 6 people (out of 7) when there are 11 clusters. A correct association means that asp and acp are 1. The last person is clustered in the 5 remaining clusters. The clustering based on the audio-video combination gives the following results (with 10 clusters): 5 (out of 7) people are correctly associated and the two remaining people are gathered in the 5 other clusters but these two participants and the not well associated person based on the video are three different persons. This means that the evolution of the clustering based on the audio-video combination is different (and better) than the evolutions based on video only or audio only. The evolution of asp when β is 0.5 is encouraging and can maybe lead to better results but the experiments are not yet done because the algorithm computation time is very long (many days). Thus by improving the weighted coefficients and doing more experiments with different values of β, we can maybe achieve a perfect multimodal clustering.

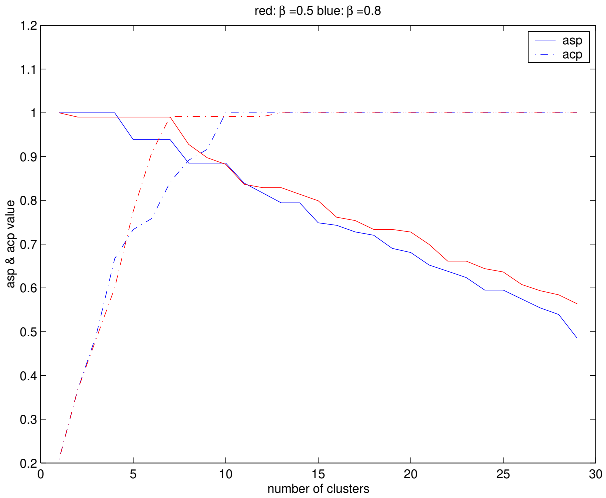


Figure 7.1: The asp and acp evolution from 6 clusters to 30. Red: $\beta = 0.5$, blue $\beta = 0.8$.

Chapter 8

Conclusions

This report presented a methodology towards speqker, face and person clustering. Given an audio-visual data stream with an unknown number of participants, we proposed a method in order to evaluate this number. A first topic is the audio clustering and it is based on an existing algorithm. A second approach relies on using images and leads to face clustering, which gives very good results but can still be improved. Finally both results are merged and a new clustering, based on audio and video scores, is computed using a hirearchical approach. As it is usually the case in person verification including faces and voices, we set more importance to the results based on the audio. Of course person clustering can also be improved by analyzing the influence of the audio results with respect to the face results on the merging choice, or by refining for instance the mixing weight coefficients.

An another interesting challenge can be the application of this method in other databases with less constraints on the behavior of the participants or even on TV-debates or news databases. In such cases, the method can not be applied without modifications: it has hence to deal with the fact that there is only one audio-video stream and the problem of face-voice association should be solved.

Appendix A

Speaker detection

A.1 Motivations

This problem is the one of data association: given a speech segmentation (found by the original speaker clustering algorithm (read 4.1)), how to associate a voice to the corresponding face? We would solve this problem by implementing a vision-based speaker detection and thus being able to extract in a meeting every speech segments of all speakers. Remember that these segments are indeed used as input of the modified *speaker clustering algorithm* (see section 4.1.2). But as the obtained results were not reliable and at the same time microphone-array-based segmentation was over, more accurate and fulfilled the same objectives, the work described in these report is not based on this studied vision-based segmentation but on a microphone-array-based one.

A.2 Audio-based speaker detection

Obviously this part should be the most important in the speaker detection. The original speaker clustering algorithm described in 4.1 should provide us with a speaker segmentation of a meeting audio file. Figure A.1 shows a theoretical example of a possible audio and video segmentation where Pi are the audio segments spoken by speaker i and Pvi are the video segments in which participant i is speaking. Note that speaker i is not necessarily participant i. The voice-face association could be done by merging the audio segmentation and a vision-based segmentation. Let us take the figure A.1 example: in this case, the task is to associate a speaker P_i and a participant P_{vj}.

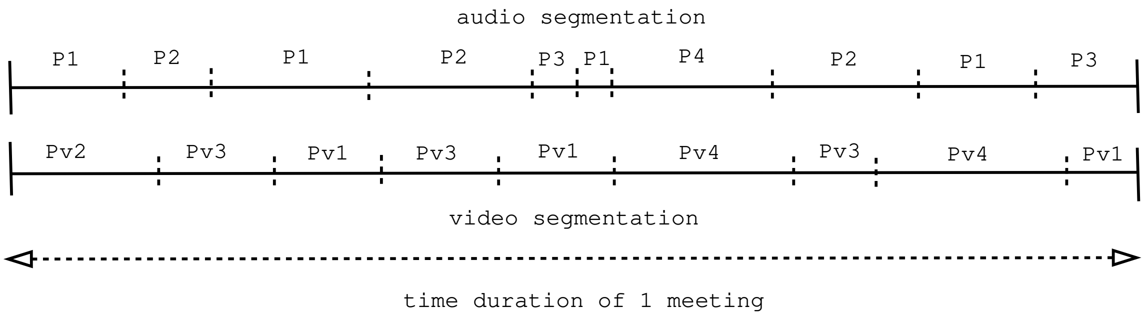


Figure A.1: Example of a theoretical audio segmentation using *speaker clustering algorithm* and a video segmentation based on the speaker detection

A.3 Vision-based speaker detection

Two main approaches could have been tested for the problem of vision-based speaker detection: the first one deals with lip motion and the second one deals with head & body motion. As the image resolution is to small, we have chosen to focus on body motion.

Thus, the vision-based speaker detection is based on the assumption that body-motion is more important when a participant is talking than when he is listening. For instance we could expect that head-motion, or arm-motion, is larger for a speaking person than for a not-speaking person.

This task can be achieved by extracting video motion features in a sub-image including the participant. Two methods have been tested but let us first define some notations:

$I(p,t)$: pixel p value of image at time t.
$\widetilde{I}(p,t)$: spatially filtered image I with a Gaussian filter of size 5x5 and $\sigma = 5/2$
$\widetilde{\widetilde{I}}(p,t)$: time filtered image $\widetilde{I}$ on 5 consecutive images (smoothing Gaussian)
(therefore it is a one-second filtering as 5 images are extracted every seconds).
Thus $\widetilde{\widetilde{I}}$ can be defined as:
$\widetilde{\widetilde{I}}(p,t) = F(1) * I(p,t-2) + F(2) * I(p,t-1) + F(3) * I(p,t) + F(4) * I(p,t+1) + F(5) * I(p,t+2)$

where $F = (0.05, 0.25, 0.4, 0.25, 0.05)$.

Pixels observations are defined as follows:

$$o(p) = \frac{\frac{1}{9}\sum_{neigh3x3} |\nabla\widetilde{\widetilde{I}}(p,t)| * |\widetilde{\widetilde{I}}(p,t) - \widetilde{\widetilde{I}}(p,t-1)|}{max(\frac{1}{9}\sum_{neigh3x3} \|\nabla\widetilde{\widetilde{I}}(p,t)\|^2, 3\text{x}3)} \tag{A.1}$$

A binary mask is then applied on the original images (see figure A.2) and features are extracted from the ROI (Region Of Interest).

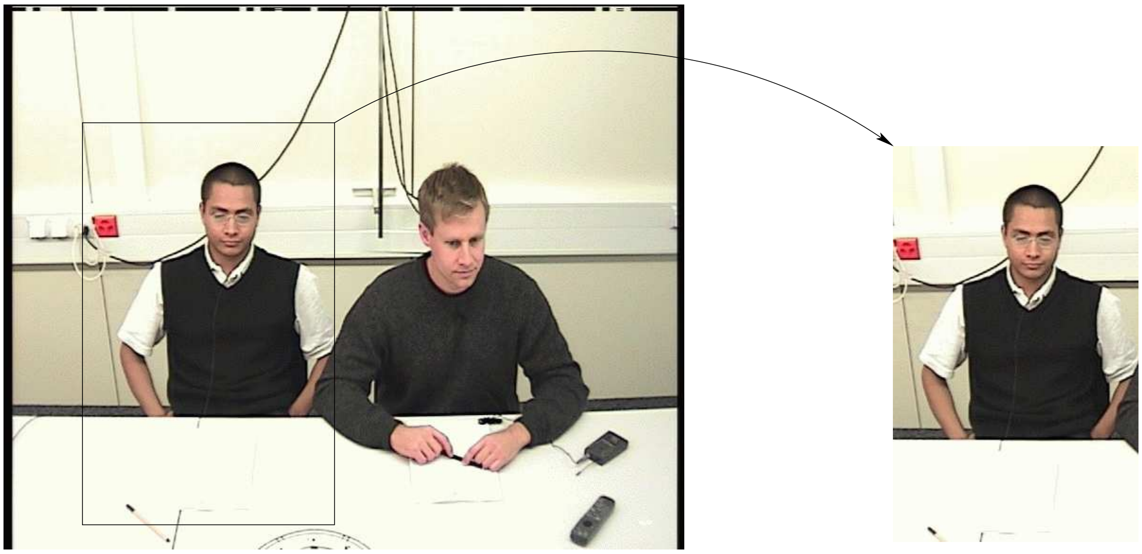


Figure A.2: Extraction of the ROI (Region Of Interest)

The two methods we have developed are based on histogram techniques and are now presented.

Method 1: The goal of this method is to find speaker independent reference feature histograms: one for a speaking-participant and one another for a listening-participant. Comparisons between the test histograms and the reference histograms should allow to make the distinction between a speaking and not speaking participant.

The features are defined as follows: sixteen features are extracted from the sub-images (see figure A.3). This should allowed to compute features values for head-motion, right-arm-motion, left-arm-motion, and so on. Each value $o(p)$ (equation A.1) of each pixel of an area are added to build a feature value. Thus each sub-image has 16 features values. An 8-bin features histogram is then computed every seconds, i.e. one histogram for 5 consecutive images.

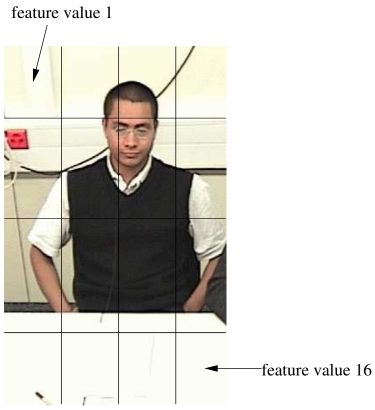


Figure A.3: Sixteen features values

The reference histograms are built in the same manner as explained above except that the interval is not one second but five seconds of one speaker and five seconds of one another. Two different participants are taken for building references in order to try to obtain good results . We chose by hand periods in which they are speaking and are not.

Comparisons between test histograms and reference histograms are then done by measuring a similarity distance. Many similarity measures could be used:

1. $\sum_{i=0}^{nbbins} \sqrt{hist[i] * href[i]}$
2. $\sum_{i=0}^{nbbins} hist[i]log(href[i])$
3. $\sum_{i=0}^{nbbins} min(href[i], hist[i])$

where $hist$ is an 8-bin test histogram, and $href$ is a reference histogram for speaking and not-speaking person. Figure A.4 shows an example of what we expect to be an ideal case.

Figure A.5 shows results based on this method using the first similarity measure and applied on one meeting. Figure A.6 shows a decision taken on these results and defining by 1 whether at time t the speaking reference histogram is higher than the not-speaking reference histogram. We know by the goundtruth that speaker 1 (Vivek) (and only him!), is speaking from 0 to 160 seconds. The results given by figure A.6 do not really match with the groundtruth.

Method 2: As the results of the first method are not reliable, we have implemented a second vision-based speaker detection algorithm.

Sub-images are not divided into 16 areas in which features values are computed but into two areas, the upper and lower part. Each pixel is defined by a couple $(o(p), \nabla I(p))$ and two cross-histograms

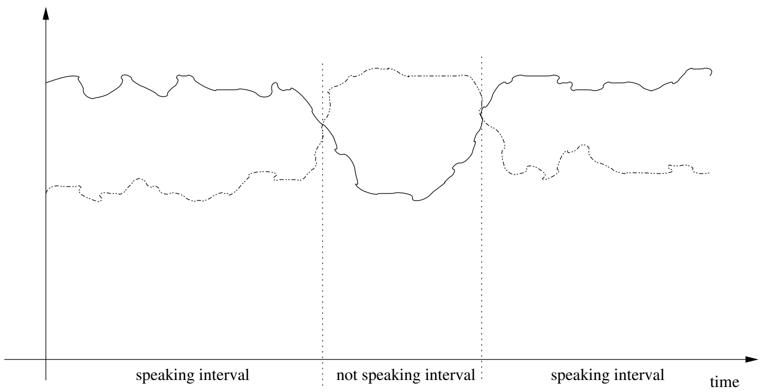


Figure A.4: Ideal case. (-) similarity measure using the speaking reference histogram; (-.) similarity measure using the not-speaking reference histogram

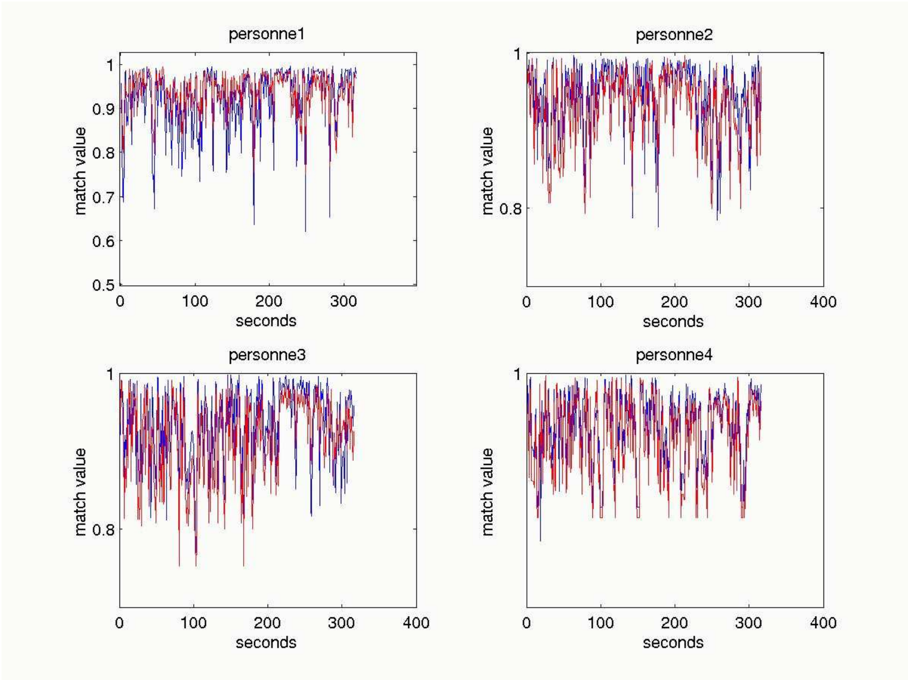


Figure A.5: Similarity measures applied on each meeting participant using: (red) speaking reference histograms; (blue) not speaking reference histograms

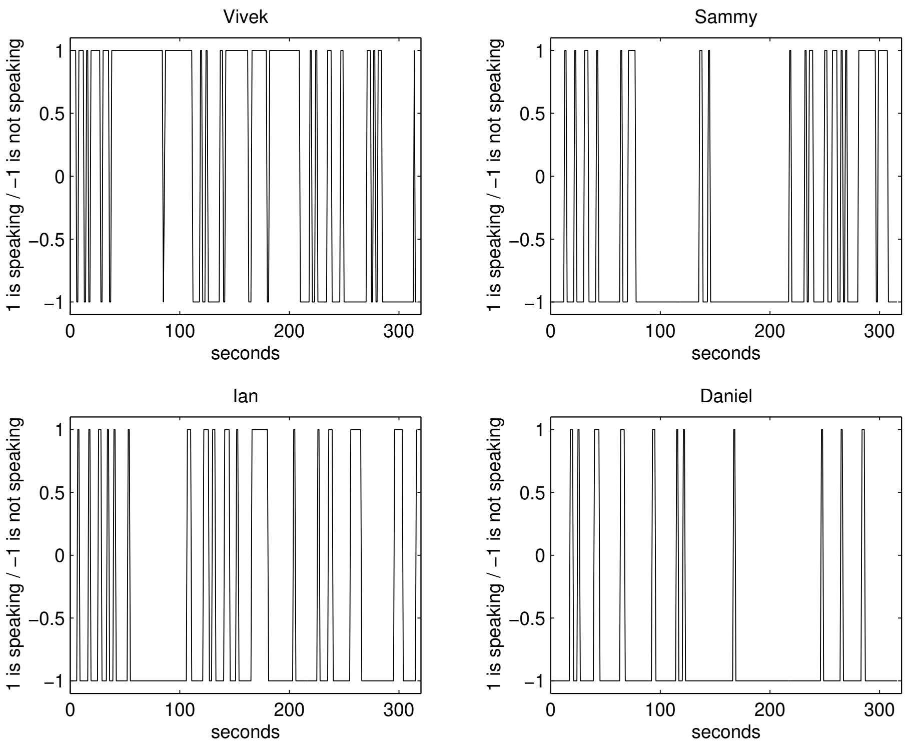


Figure A.6: Decision based on the score of speaking and not speaking reference histograms per sub-images are computed based on these couples. Observations $o(p)$ are binned into 7 values and gradient $\nabla I(p)$ into 4 values.

Let us define the histogram mean and variance:

$$\overline{m} = \frac{\sum_{i=0}^{nbbins} hist[i]*center[i]}{\sum_{i=0}^{nbbins} hist[i]}$$
$$\sigma^2 = \frac{\sum_{i=0}^{nbbins} hist[i]*center[i]^2}{\sum_{i=0}^{nbbins}} - \overline{m}^2$$

where *hist* is the test histogram and *center* are the bins center.

We expected to have *peaks* values either for mean or for variance when *hist* corresponds to an histogram of a speaking period.

Figure A.7 shows the mean and the variance values for one meeting participant. Blue line is mean and variance of sub-image upper part, and red line is for the lower part. Remember that the third person is not speaking from 0 to 160 seconds. The results of other people are not displayed in this report but are similar the given ones and, based on the studied meeting, are not enough reliable.

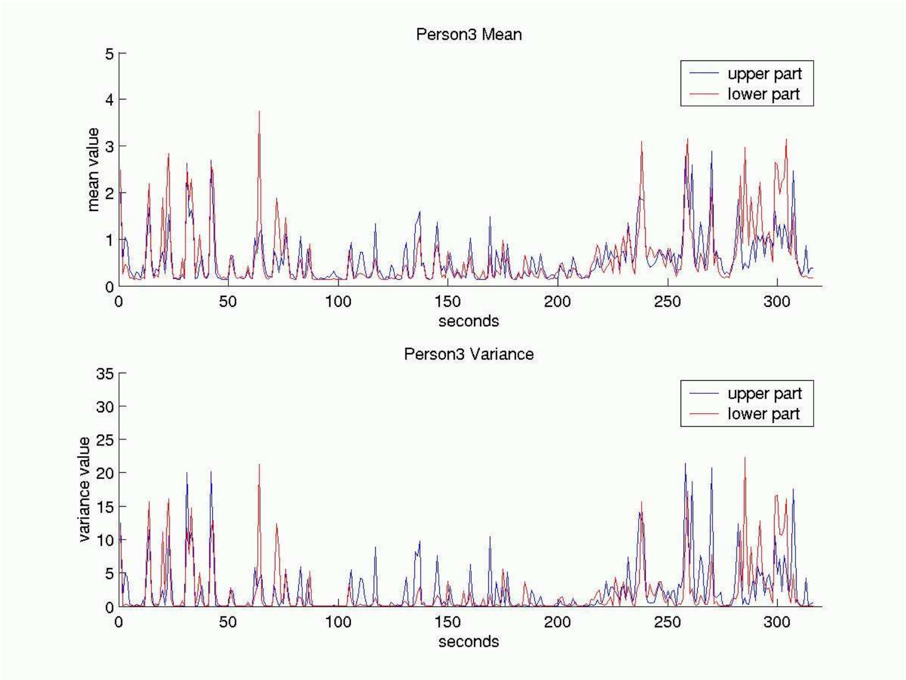


Figure A.7: mean and variance values of both parts of the extracted sub-images for one meeting participant

Conclusions: The tested methods do not give reliable results and we can doubt of their efficiency. They were however tested on only one meeting and conclusions can not be drawn without further experiments. Moreover microphone-array-based segmentation gives 3D spatial information allowing a reliable and accurate voice-face association and has therefore been exploited for further work.

Bibliography

[1] Darren C. Moore. *The IDIAP Smart Meeting Room.* IDIAP Communication report, Martigny, November 2002.

[2] Jianbo Shi & Jitendra Malik. *Normalized Cuts and Image Segmentation.*

[3] Pavel Berkhin. *Survey of Clustering Data Mining Techniques.*

[4] L. Kaufman & P. Rousseeuw. *finding groups into data: an introduction to cluster analysis.* John Wiley and Sons, New York, 1990.

[5] R.Ng & J.Han. *"Efficient and effective clustering methods for spatial data mining", in Twientieth Intenational Conference On Very Large Database.* pp 144-155, Morgan Kaufmann, 1994.

[6] A.Jain & R.Dubes. *Algorithms for Clustering Data.* Prentice Hall, Englewood Cliffs, NJ, 1988.

[7] J.Hartigan & M.Wong. *Algorithm AS136: A k-means clustering algorithm.* pp 100-108, Applied Statistics, 1979.

[8] T. Mitchell. *Machine Learning.* McGraw-Hill, New York, NY, 1997.

[9] J-M Odobez & P. Bouthemy. *Estimation robuste muti-échelle de modèles paramétrisés de mouvement sur des scènes complexes.* AFCET-RFIA, Paris, January 1994.

[10] A. Fitzgibbon & A. Zisserman. *On Affine Invariant Clustering and Automatic Cast Listing in Movies.* Proceedings, ECCV, 2002.

[11] A. Jitendra & H. Bourlard & I. Lapidot & I. McCowan. *Unknown-Multiple Speaker Clustering unsing HMM.* IDIAP Research Report, April, 2002.

[12] R. O. Duda & P. E. Hart & D. G. Stork. *Pattern Classification.* pp 550-553, Whiley Interscience, second edition, 2001.

[13] S-C Chu & J. F. Roddick & J. S. Pan. *Efficient K-Medoids Algorithm Using Multi-Centroids With Multi-Runs Sampling Scheme.* 2001.

[14] P. Berkhin *Survey of Clustering Data Mining Techniques.* Accrue Software, Inc, San Jose 2002.

[15] E. Hjelmas, B. K. Low. *Face Detection: A Survey.* Computer Vision and Image Understanding 83,236,274, 2001.