IDIAP COMMUNICATION

ALGORITHMS FOR VIDEO STRUCTURING

Maël Guillemot [a b c] Jean-Marc Odobez [a]
Daniel Gatica [a]

IDIAP–COM 02-05

30TH OF AUGUST, 2002

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] IDIAP, CP 592, 1920 Martigny, Switzerland
[b] IFSIC, Université de Rennes 1, Rennes, France
[c] http://www.idiap.ch/ guillemo

First of all, I would like to thank Dr. Jean-Marc Odobez for the opportunity he gave me to do this internship at the Dalle Molle Institut for Perceptual Artificial Intelligence, as an introduction to the research world. I also would like to thank him for all his supervision,

A special acknoledgement is addressed to Dr. Daniel Gatica-Perez who gave me instructions and advices all along the project,

The Eastman Kodak Company (C) is acknoledged for providing the Home Video Database,

I would like also here to thank Dr. Luce Morin[1] *for the chance she gave me to study as an Erasmus student in the UPC*[2] *during the semester Sept. 01 - Feb. 02.*

I would also like to thank everybody from the Vision Group including Alessandro, Beat, Cédric, Datong, Fabien, Florent, François, Guillaume, Himanshu, Mark, Sébastien and other Idiap students, Alexeɨ, Andrew, Guillaume, Haɨang, Jaume, Quan, Mohamed, Norman, Simon for being friendly and welcoming, the very anonymous english reviewers Christos, Conrad, Mark, Todd and Matthew for his technical help in using LaTeX ,

For the official paperwork, lodgement and the logistic issue, I would like to thank Nadine Rousseau and Sylvie Millius for being ever-ready and efficient in dealing with the administration.

[1] INRIA/IRISA, Campus de Beaulieu, Rennes, France

[2] Universitat Politecnica de Catalunya, Barcelona, Spain

ABSTRACT

Video structuring aims at automatically finding structure in a video sequence. Occupying a key-position within video analysis, it is a fundamental step for quality indexing and browsing. As a low level video analysis, video structuring can be seen as a serial process which includes (i) shot boundary detection, (ii) video shot feature extraction and (iii) video shot clustering. The resulting analysis serves as the base for higher level processing such as content-based image retrieval or semantic indexing. In this study, the whole process is examined and implemented. Two shot boundary detectors based on motion estimation and color distribution analysis are designed. Based on recent advances in machine learning, a novel technique for video shot clustering is presented. Typical approaches for segmenting and clustering shots use graph analysis, with split and merge algorithms for finding subgraphs corresponding to different scenes. In this work, the clustering algorithm is based on a spectral method which has proven its efficiency in still-image segmentation. This technique clusters points (in our case features extracted from video shots) using eigenvectors of matrices derived from data. Relevant data depends of the quality of feature extraction. After stating the main problems of video structuring, solutions are proposed defining an heuristical distance metric for similarity between shots. We combine color visual features with time constraints. The entire process of video structuring is tested on a ten hours home video database.

RÉSUMÉ

La structuration de séquences vidéo occupe une position clé dans le processus général d'analyse vidéo, en amont des traitements plus haut niveaux telles que l'indexation et la navigation. La structuration de vidéo peut être vue comme un processus séquentiel qui inclus (i) la détection de changement de plans, (ii) l'extraction de descripteurs des différents plans vidéo et (iii) le regroupement des plans en scènes. Dans l'étude que nous proposons, le processus global est examiné et implémenté. Deux détecteurs basés respectivement sur l'estimation du mouvement et sur l'analyse de la distribution de couleur sont développés. L'extraction de descripteurs caractérisant chaque plan vidéo est une étape importante pour la classification. Les approches traditionnelles de classifications regroupent des points en s'inspirant de la théorie des graphes pour repérer les différents sous-graphes correspondant aux différentes scènes. Dans l'étude que nous proposons, une nouvelle technique de classification est présentée basée sur des avancées récentes en apprentissage artificiel et reconnaissance de formes. L'algorithme de classification est basé sur l'analyse spectrale - des valeurs propres - de la matrice d'inter-distance. Cette méthode a prouvée son efficacité en segmentation spatiale d'images. Des solutions aux principaux problèmes de structuration de séquences vidéo sont proposées et analysées en définissant une distance de similarité entre plans vidéo. Enfin, les algorithmes proposés ont été testés et évalués sur une base de données de vidéo 'grand public' de dix heures.

Contents

Chapter 1

Introduction

General introduction to video / multimedia A video is defined as a sequence of images. This study is thus related to one part of the audio-visual analysis which focuses both on audio and video sequences. The technology market trends are clearly towards an increase in the use of images and the audio-visual sequences. With the explosion of communication through internet, and the accessible price of digital cameras and video cams, everybody can archive multimedia data. Therefore, the number of home video databases has increased as well as family photo albums. TV programs are another large source of video databases that could be automatically classified into movie/drama, news/current affairs, sport, youth programs, documentaries, etc. In the professional area, a possible need is a system which enables structuring, browsing and querying of an archive of automatically annotated meetings. One can think also of applications such as security management. An important point is to organize these data to enable retrieving and fast browsing. All these large databases are expected to be semantically summarized. Here is a brief review of important research conducted in the area of video analysis:

- Automatic video indexing based on image, text, and audio content
- Robust person and text detection and recognition
- Road traffic, transport, security, infra-red
- Weather forecasting, satellite images, sonar-acoustic images
- Medical images/video, biological modeling, endoscopy, robot vision, experimental visualization in fluid mechanic
- Event detection, recognition and understanding, cross media content extraction

Researchers in this area have recently grouped their skills into a standard, MPEG-7 (multimedia content description standard), and this standard is poised to revolutionize multimedia content management. The MPEG-7 standard provides a standardized meta-data system for describing multimedia content. MPEG-7 allows inter-operable indexing, searching, and retrieval of video, images, audio, and other forms of multimedia data. It also allows for very rich description of multimedia content, including features, structure, semantics, models and collections.

What is video structuring and why do it? Video shot structuring is the problem of automatically finding structure in a video sequence. Research has focused for the last fifteen years on the automatic analysis of videos.

Video indexing needs to get as an input the organization of the video: the structure. Typically the indexing of a video sequence consists of the following steps: segment the video hierarchically into sequences, scenes, and shots where a shot is a continuous sequence of frames captured from one camera (useful terms are defined in Chapter 4). A scene is composed of one or more shots which present different views of the same event, related in time or space. A segment is composed of one or more related scenes. Possible features extracted from a video are bibliographic information (title, creator, dates, subjects, item numbers, publisher details, names, synopsis etc.), format, framerate, duration etc . Indexing video analysis [**?**], content-based image query retrieval (no text) implies the need for semantic similarity comparison. In this report, the major problems that we are dealing with are the characterization of the shot changes, the representation of data by defining similarity measures, and a method for clustering.

Usefulness of this work for the hosting institute The software I developed will be used in the future for different projects. As an example of fast browsing, let's imagine that I (as a user) want to find the shots where I was going for a wonderful hike in the mountains. As a first step before applying higher level processes, the designed software can be used for object, text or event detection inside a video. Also having a large image representing the entire video with shot key-frames can be useful.

IDIAP is currently involved in two European projects in this field:

- $ASSAVID$, Automatic Segmentation and Semantic Annotation of Sports Videos. The aim of this project is to develop techniques for automatic segmentation and semantic annotation of sports videos.
- M^4, Multi-Modal Meeting Manager. This project is concerned with the construction of a demonstration system to enable structuring, browsing and querying of an archive of automatically analyzed meetings. The archived meetings will take place in a room equipped with multimodal sensors.

Plan of this report Before going deeply through the project on its own, Chapter 2 presents the hosting institute in Switzerland and particularly the computer vision group inside which I worked. Chapter 3 examines the project definition in terms of existing material and goals. Video structuring can be seen as a serial process: a video is naturally segmented into a succession of shots. Thus the first step of this process is to detect the shot boundaries. Several techniques exist and chapter 4 gives a brief review of the state-of-the-art and details two methods implemented during the project. Once the video has been segmented we want to group the shots that have a same "meaning" together. In order to do this, video shot features are extracted so as to compare features between different video shots within the video. Chapter 5 discusses the feature extraction process. Three main algorithms are detailed. The last step of the serial process of video structuring is the heart of this project: video shot clustering (or grouping shots that are similar into scenes). In chapter 6, typical approaches for shot clustering are first discussed and a novel technique namely spectral video structuring is examined and experiments are computed on a ten hours home video database. Chapter 7 draws some concluding remarks and give ideas for further work.

Chapter 2

About the hosting institute

2.1 Presentation and Research activities

Created in 1991 by the Dalle Molle Foundation for the Quality of Life, the Dalle Molle Institute for Perceptual Artificial Intelligence (IDIAP[1]), located in Martigny (Valais, Switzerland), is a not-for-profit research institute affiliated with the Swiss Federal Institute of Technology in Lausanne (EPFL) and the University of Geneva.

In 2001, IDIAP numbered an average of 40-45 collaborators, including permanent scientific staff, post-doctors, PhD students (around 18), system and development engineers, and short-term to medium-term visitors.

The activities carried out at IDIAP can be described as follows: research and development activities, participation in European and national research projects, collaborations with organizations and companies, and teaching and training activities. IDIAP's mission therefore consists in:

- Carrying out fundamental and applied research activities aiming at long and medium term industrial transfer.
- Teaching and training activities.

IDIAP is divided into three research areas (as illustrated in figure 2.1: machine learning (sequence processing, Neural networks, HMM), Speech (linguistic modeling of spoken language, speaker authentification, speech generation and synthesis) and Vision (described bellow).

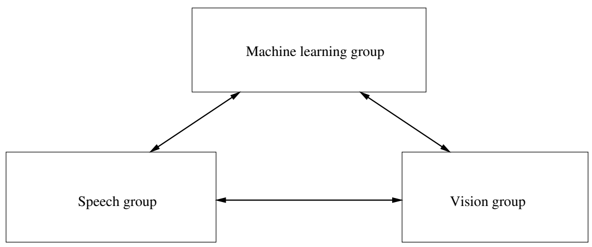


Figure 2.1: Idiap Research groups

[1] http://www.idiap.ch

2.2 Computer Vision Group

The computer vision group studies problems in machine visual perception, such as media annotation, people detection and human gesture tracking and recognition. Research activities center on multi modal interpretation of visual and multimedia data, and improvement of basic detection and classification measures and algorithms. This improvement may be achieved by enhancing and extending existing algorithms, or by creating new algorithms and measures. This frequently involves collaboration across research groups, as complementary expertise is brought to bear on a problem.

There is strong expertise within the vision group in areas of text processing from both documents and video, object tracking and recognition of gesture, and domain based video annotation. The group is active in all of these areas under a number of collaborative European and Swiss national projects.

1. Document Analysis and Recognition: Work in this area involves applying non-traditional methods to improve both preprocessing and recognition steps. In particular, methods from the speech processing area are being examined for use to improve recognition.

2. Text Detection and Recognition in Images and Videos: The vision group is involved in text detection and segmentation algorithms, and also examination of new paradigms in video text recognition. The goal of current research is to reduce false positive detection, and move away from explicit segmentation as a preprocessing step.

3. Face Algorithms: Face algorithm can be divided into four different areas.

   - Face detection : The goal of face detection is to identify and locate human faces in images at different positions, scales, orientations and lighting conditions.

   - Face localization : Face localization is a simplified face detection problem with the assumption that the image contain only one face.

   - Face verification : Face verification is concerned with validating a claimed identity based on the image of its face, and either accepting or rejecting the identity claim.

   - Face recognition : The goal of face recognition is to identify a person based on the image of its face. This face image has to be compared with all the registered persons. Therefore, face recognition is computationally expensive regarding the number of registered persons.

New research area: Image and video annotation The purpose of image and video annotation is to provide access to the ever increasing digital archives of such data. Whether these archives are within a television station or publicly available web documents, the sheer volume of data being produced at any moment is beyond human ability to annotate. In addition there are large historical archives that contain priceless data recording important moments. Television stations will use such technology to provide a method of access to their archives, such as sports and news, and to access historical footage to enrich current programs, and for documentary pieces. Multimedia annotation involves both visual and audio data, and the deduction of higher level, semantic annotation which must be conducted under a machine learning framework. The work in this area thus brings together all three groups at IDIAP. Accretive annotation uses low level feature information from a number of modalities, such as audio and video, to work towards semantic annotation. For this work domain specific information is used to provide a framework for interpreting the low level data, to direct progressively higher level processing for increasingly detailed annotation.

Chapter 3

Project definition

Origin and nature of the project This work takes place in the context of new research activities of IDIAP in video analysis and image indexing. One of these new research interests is content-based retrieval, that is on its own a vast area and a challenging task: the aim is to be able to retrieve semantically meaningful information using similarity between visual, spatial, temporal and motion attributes. Video structuring as illustrated in figure 3.1 is composed of three major technological bricks and is an fundamental step before higher level video analysis processes.

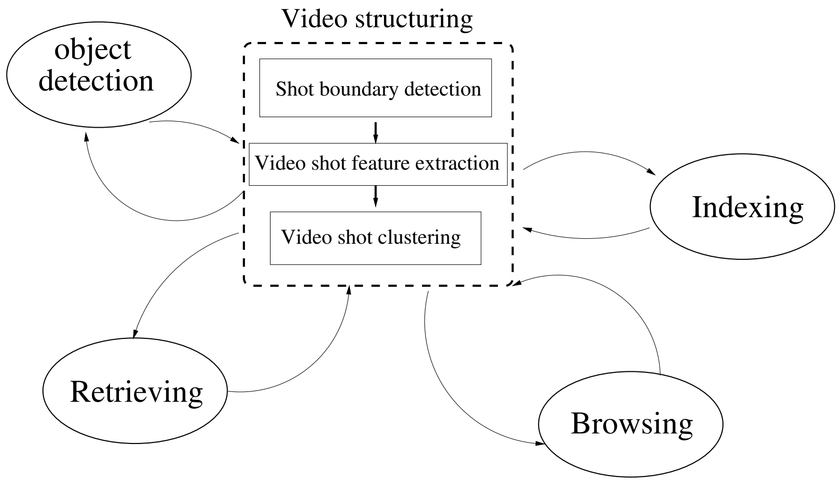


Figure 3.1: Around video structuring

Our work takes place in the context of the two european projects described in Chapter 1 and will be useful for any other tasks dealing with text detection, event detection or object recognition in a video sequence.

3.1 Existing material inside the laboratory

- Softwares

  A software capable of extracting global motion parameter ζ (see Chapter 4) was already developped. Dr Jean-Marc Odobez did investigation in the area of motion estimation in the past

([?]). A new library was installed during the first month of my project: OpenCV[1] is a computer vision library for extracting and processing meaningful data from images, see for more information. A higher level object-oriented slice have been added by my two supervisors in order to read ppm files from video and to save them as a 'home-made' image object derived from OpenCV structure. Dr. Daniel Gatica-Perez did investigation on home video structuring using probabilistic approach [?]. Software outputing statistical results of shot clustering performance evaluation was also exploited.

- Video databases

  The first database is from the $ASSAVID$ project briefly discribed in introduction. This database consists of sport video from Barcelona olympic games (tennis, football, etc.). Only football videos have been exploited from this database.

  The novel clustering algorithm have been implemented over a ten hours $Kodak(c)$ home video database in R-G-B format. A detailed description can be found in section 6.3. A third part ground truth for each video of this database was delivered.

3.2 Main objectives of the project

- The first main objective of this internship is to develop several tools for temporal segmentation and video shot clustering, using motion - color - form under Linux / Unix. These implementation should be at the state-of-the-art in the research area of video analysis.
- The second objective is to test and compare different methods for shot clustering (typical approaches, hierarchical clustering with hypothesis tests and spectral methods).

3.3 Project scheduling

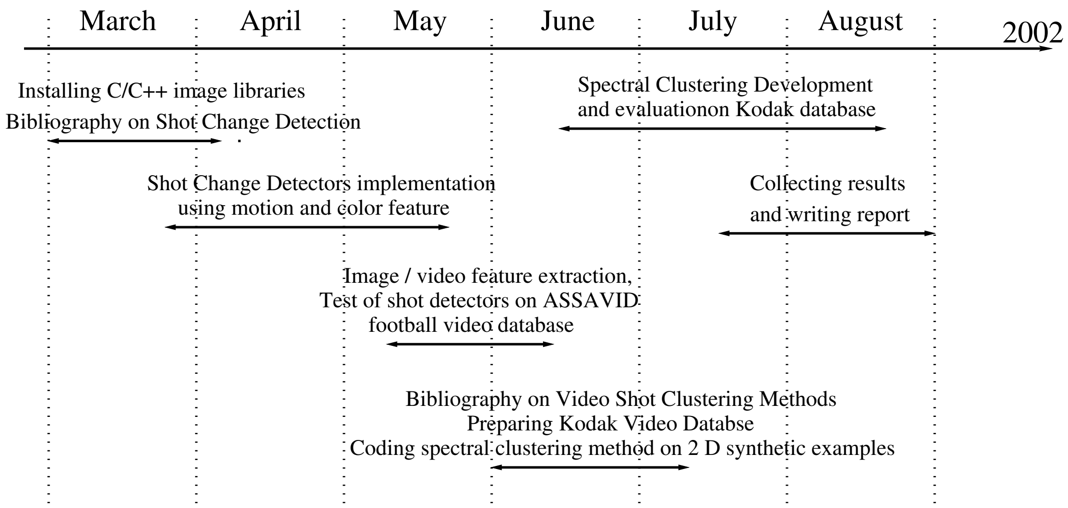


Figure 3.2: Intern-ship scheduling

[1]Intel (c) Open Source Computer Vision Library

Chapter 4

Détection de changements de plans dans une séquence vidéo

Détecter les changements de plans dans une vidéo permet de découper temporellement une séquence vidéo. Le développement de ces techniques a fait l'objet de nombreuses recherches lors de ces dix dernières années. Les principales approches sont définies et comparées dans ce chapitre. Les deux méthodes développées sont ensuite étudiées: nous présentons la modélisation de type -objet- du programme. La dernière section présentent le résultat de leurs performances sur une séquence vidéo de football.

4.1 État de l'art

La segmentation temporelle d'une séquence vidéo en plan est le premier traitement à effectuer comme précédemment illustré (figure 3.1). On cherche à détecter particulièrement les *frontières* des plans vidéo (début et fin) et le type de transitions entre ceux-ci. Une fois le signal vidéo découpé en une collection de segments - plans -, ces élements sont utilisés comme briques de bases pour le regroupement en scène (Chapitre 5 et 6) puis - plus haut-niveau encore - pour l'indexation.

Les principaux termes techniques sont définis dans la table 4.1. Les trois derniers font référence aux transitions les plus utilisées.

plan /shot	une séquence d'images ininterrompues provenant d'une même caméra
image-clé / key-frame	une image représentative du plan ou sous-plan
scène / scene	une collection d'un ou plusieurs plans consécutifs liés à un ou plusieurs objets d'intérêt
balayage / Pan - Tilt	balayage horizontal/vertical de la caméra
coupure / cut	un changement abrupt d'une image à l'autre
fondu / dissolve	l'image du premier plan s'obscurcit tandis que l'image du deuxième plan apparait de plus en plus claire
volet / wipe	transition lente où les pixels du 2ème plan remplace ceux du 1er d'une façon régulière

Table 4.1: Definition de mots-clés dans le domaine de audio-visuel

Les études [?] et [?] proposent deux études comparatives détaillées de détecteurs de changements de plans vidéo. Dans les cinq algorithmes décris, on retrouve toujours le même schéma général (figure 4.1).

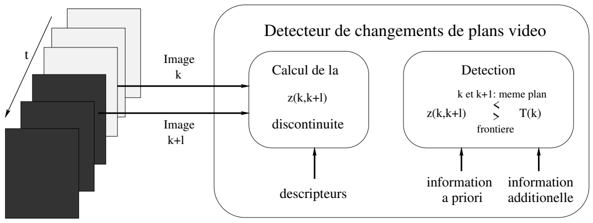


Figure 4.1: Détection de changements de plans

À partir de caractéristiques extraites, une mesure calcule une discontinuité entre deux images. Cette discontinuité est ensuite comparée à un seuil -adaptatif ou non- pour décider oui ou non si un changement a lieu à cet instant.

Plus formellement, on peut poser le problème ainsi: deux images k et $k+l$ sont comparées à l'aide d'une mesure de dissimilarité $z(k, k+l)$. Sa valeur permet de savoir si ces deux images font parti d'un même plan S ou pas. Soit τ un seuil:

$$\begin{cases} (z\,(k, k+1) \geq \tau) \;\Rightarrow\; k \in S,\, k+1 \in S+1 \\ \quad (z\,(k, k+1) < \tau) \;\Rightarrow\; (k \bigcup k+1) \in S \end{cases} \tag{4.1}$$

La séquence d'image est donc associée à la variable z évoluant au cours du temps. Les approches décrites ci-après donnent différentes façons de définir une mesure de similarité.

Comparaison pixel/pixel Un pixel (**pic**ture **el**ement) est la plus petite unité d'une image numérique. La façon la plus simple de mesurer la similarité entre deux images est donc de calculer la somme des différences de niveaux de gris -ou couleur- d'un pixel $I(x, y) \in \aleph$ dans l'image k au même pixel dans l'image $k+1$ (en valeur absolue), Soit:

$$z(k, k+1) = \frac{\sum_{x=1}^{X} \sum_{y=1}^{Y} \sum_{c} |I_k(x, y, c) - I_{k+1}(x, y, c)|}{X \cdot Y} \tag{4.2}$$

où X (respectivement Y) désigne le nombre de colonnes (respectivement de lignes) dans l'image et c, l'indice de la composante de couleur (c = 1 pour une image en niveaux de gris)

Cette méthode a un sérieux inconvénient: au moindre mouvement de caméra ou d'objet spatialement important au cours d'un plan, cette technique détecte un changement de plan. Une amélioration possible de cette méthode est de compter le nombre de pixel qui change de valeur plus "vite" qu'un certain seuil et de comparer ce nombre avec un second seuil. Néanmoins, même si cette technique est rapide, elle reste sensible aux mouvements (caméra ou objet(s)).

Approche par differences statistiques Les methodes stastiques sont basées sur l'approche prédemment décrite en découpant spatialement chaque image en L régions et en comparant des mesures statistiques des pixels dans ces régions. Soit ν une variable aléatoire discrète associée à un niveau de gris $I(x, y) \in \aleph$, pour chaque bloc l de l'image k.

Chaque image k est divisée en L blocs et comparée avec le bloc correspondant dans l'image $k+1$. Typiquement, la différence entre les images k et $k+1$ est mesurée par:

$$z(k,k+1) = \sum_{l=1}^{L} c_k \cdot z_0(k,k+1,l) \tag{4.3}$$

où c_k est un poids connu a priori du bloc k et $z_0(k,k+1,l)$ est une valeur de correspondance partielle entre les $l$ièmes blocs des images k et $k+1$.

Un exemple de $z_0(k,k+1,l)$ peut être défini comme telle:

$$z_0(k,k+1,l) = \frac{\left[\left(\frac{\sigma_{l,k}(\nu)+\sigma_{l,k+1}(\nu)}{2}\right)^2 + \left(\frac{\mu_{l,k}(\nu)-\mu_{l,k+1}(\nu)}{2}\right)^2\right]^2}{\sigma_{l,k}(\nu)\cdot\sigma_{l,k+1}(\nu)} \tag{4.4}$$

où μ est la moyenne et σ, l'écart type de la variable ν.

Ces techniques utilisent des characteristiques spatiales locales pour augmenter la robustesse aux mouvement de camera et d'objets.

Approche par analyse de la distribution de couleurs: l'histogramme Les méthodes basées sur l'analyse d'histogrammes utilisent la distribution de couleurs ou de luminance globalement ou localement (en divisant l'image en 16 régions par exemple). En contraste à la méthode précédente qui utilise des variables caractéristiques, ici, on travaille à partir de la distribution entière. Un histogramme normalisé correspond à la fonction de distribution de probabilité (PDF en anglais) $H(j) = PDF\{I(x,y)\}$, où $I(x,y)$ est l'image en niveau de gris ou un composant RGB d'une image couleur et j est l'index du niveau de gris. La figure 4.2 tirée de la base de donnée grand public montre trois histogrammes rouge, vert et bleu.

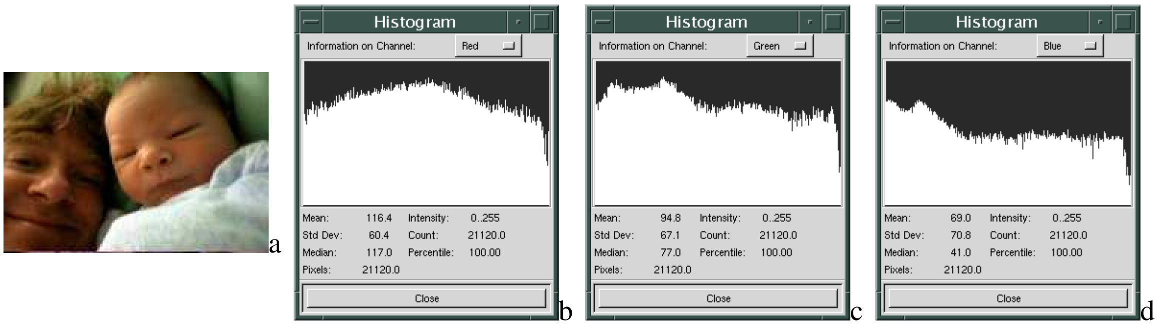


Figure 4.2: Distribution de couleur dans une image (a) histogrammes rouge, vert, bleu (b,c,d)

En pratique, pour diminuer le temps de calcul, le nombre d'index de niveaux de gris est diminué. À chaque index (de 0 à 15 par exemple), correspond une mesure de probabilité qu'un évenement (dans notre cas le niveau de gris d'un pixel) tombe dans la gamme donnée de la variable discrète k. On souhaite trouver la mesure de similarité $z(k,k+1)$ entre les deux histogrammes H_k et H_{k+1}. De la même manière que les autres méthodes, une transition est détectée quand la mesure de dissimilarité est plus élevée qu'un certain seuil prédéfini τ. On trouve plusieurs distances dans la literrature:

- Distance L1:

$$z(k, k+1) = \sum_{j=0}^{n} |H_k(j) - H_{k+1}(j)| \tag{4.5}$$

  où j peut soit être un niveau de gris soit un indice regroupant plusieurs niveaux de gris (pour traiter moins d'information), n étant le maximum de niveaux représentés.

- Intersection d'istogramme:

$$z(k, k+1) = 1 - Intersection(H_k(j), H_{k+1}(j)) = 1 - \frac{\sum_{k=1}^{n} \min(H_k(j) - H_{k+1}(j))}{\sum_{k=1}^{n} \max(H_k(j), H_{k+1}(j))} \tag{4.6}$$

- Test de χ^2

  Aussi, d'autres mesures de dissimilarité sont calculées comme le "taux de vraissemblance", le test de Kolmogorov-Smirnov etc. Par exemple, le test de χ^2 compare deux histogrammes $H_k(j)$ et $H_{k+1}(j)$ de deux images consecutives k et $k+1$:

$$z(k, k+1) = \sum_{j=0}^{n} \frac{|H_k(j) - H_{k+1}(j)|^2}{H_{k+1}(j)} \tag{4.7}$$

On peut appliqué un prétraitement sur la courbe z. Par exemple, [?] a appliqué un algorithme de morphologie mathématique 1D, noté *chapeau haut de forme* afin d'en extraire les pics hauts et étroits:

$$C_k = Chapeau(z_k) = z_{k-1} - Ouvert(z_{k-1}) = z_{k-1} - Dilate[Erode(z_{k-1})] \tag{4.8}$$

où Ouvertz_{n-1} représente l'ouverture morphologique, composée d'une érosion suivie d'une dilatation.

L'analyse d'histogramme est la technique la plus utilisée pour la détection de changement de plan vidéo. Elle offre l'avantage d'être invariante aux changements tels que la rotation d'une image ou les changements d'échelles (zooms par exemple). Par contre, deux images complètement différentes peuvent peuvent avoir le même histogramme.

Comparaison des coefficients DCT Cette approche est basée sur la différence des coefficients DCT (transformée de Fourier discrète) de bloc compressé (8*8) comme c'est le cas dans le codage JPEG. Le premier coefficient de chaque bloc est comparé avec celui du mêmne bloc dans l'image suivante. C'est en effet le coefficient qui a le plus d'information: la valeur moyenne du bloc.

Cette methode a le principal inconvénient de ne pas être rapide, à moins que l'on ne manipule des image déjà compressées.

Comparaison des contours d'une image Le pourcentage des contours (c'est à dire les pixels sur les côtés de l'images) qui "entrent et sortent" entre deux images consécutives sont calculées. Un changement de plan est détecté quand le pourcentage des changements de contours est élevé. Les transitions graduelles telles que les fondus sont identifiées mais cette méthode restent très sensible au mouvement.

Approche par estimation du mouvement L'idée principale de cette approche est d'estimer le mouvement entre deux images. L'analyse des caractéristiques d'estimation de mouvement permet à la fois de détecter le type de transition (coupure, fondu, volet, etc.), et certains effets comme un zoom par exemple. Cette technique permet donc d'extraire plus d'informations que les autres précédemment citées. Plusieurs manières d'estimer le mouvement existent dans la literrature. La plus connue est sans doute la mise en correspondance de blocs: "bloc matching" en anglais. On retrouve cette méthode dans la norme de codage vidéo MPEG. Brièvement, l'idée est de découper chaque image en bloc et de rechercher dans l'image suivante où chaque bloc se retrouve. Ainsi, on peut estimer le mouvement global de la caméra et/ou le mouvement local des objets dans la vidéo. Cette méthode revient cher en terme de calcul mais est très efficace.

Une autre technique est décrite dans la section suivante qui présente les détecteurs implémentés au cours de ce stage.

4.2 Les deux approches développées

La détection par analyse d'histogrammes est très communément utilisée en raison de sa simplicité et de sa rapidité d'exécution. En général, les algorithmes simples ont de meilleurs performances que les plus compliqués. Par exemple, un algorithme utilisant une analyse par histogramme dans chaque bloc d'une image n'a pas forcément de meilleurs résultats que l'analyse par histogramme global. Cette section examine les deux techniques utilisées: par estimation de mouvement d'une part et par analyse d'histogramme global d'autre part.

Detection par estimation de mouvement global motion paramete Ce premier détecteur de changement de plan a été inspiré de [**?**]. Cette methode exploite l'information de mouvement entre deux images. Le mouvement dominant est représenté à chaque instant par un modèle affine. Plusieurs études sur ce sujet existent, [**?**] en est une. Les auteurs ont choisi un modèle affine w_Θ défini à la position du pixel $p = (x, y)$, considérant un point de référence (x_g, y_g) (4.9).

$$w_\Theta(p) = \begin{pmatrix} a_1 + a_2(x - x_g) + a_3(y - y_g) \\ a_4 + a_5(x - x_g) + a_6(y - y_g) \end{pmatrix} \tag{4.9}$$

Ce modèle affine permet de modéliser des mouvements tels que les balayages, les zooms, les déplacements spécifiques d'objets, etc. Un 7ème paramètre η est estimé, qui est la variation global d'intensité inter-image. Le vector de paramètres $\Theta = (a_1, a_2, a_3, a_4, a_5, a_6, \eta)$ est estimé entre deux images k and $k + 1$.

Soit DFD (*displaced frame difference*):

$$\boxed{DFD_\Theta(p, d_\Theta = I_{k+1}(p + d_\Theta(p)) - I_k(p) - \eta \qquad (4.10)}$$

Notons que dans la formule (4.10), on retrouve la même structure que dans l'équation (4.5). La nette différence ici est que l'on tient compte **du mouvement interne** possible entre images successives d'une vidéo. On estime alors Θ et η par la méthode décrite dans [**?**].

On définit $Grille$ comme le cadre de l'image et E l'ensemble des points (c'est à dire les pixels) tels que $p \in Grille \wedge p + w_{\hat{\Theta}}(p) \in Grille$. Soit ζ, une variable extraite des paramètres définis précédemment et s, un seuil indiquant un écart maximum en niveaux de gris/couleurs. On impose alors à DFD la contrainte suivante:

$$\begin{cases} n_1 = Card(E) \\ n_2 = Card(p \in E \ / \ |DFD(p, d_{\hat{\Theta}})| < s) \end{cases} \tag{4.11}$$

$$\zeta = \frac{n_2}{n_1} \tag{4.12}$$

La variable ζ est donc un rapport entre n_1, le nombre ($Card$) de pixels qui reste à l'intérieur du cadre de l'image ($Grille$) après déplacement et n_2 le nombre de pixels qui, après déplacement dans l'image suivante, n'a pas changé abruptement de niveau de gris (ou couleur). ζ nous donne de l'information sur le mouvement dominant entre deux images consécutives.

Test de Hinkley Le point important maintenant est de définir un critère approprié pour valider les sauts de la variable ζ_t parmis de faibles variations impertinentes. Les auteurs emploient un test appelé le test de Hinkley (venant du domaine de l'automatique), connu pour être robuste en prenant en compte "tout le passé" de la quantité observée. Deux séries de calculs et de tests sont en fait effectués en parallèle pour chercher les sauts vers le bas et vers le haut. Ils sont respectivement définis par 4.2.

$S_k = \sum_{t=0}^{k}(\zeta_t - m_0 + \frac{\delta_{min}}{2}), (k \geq 0)$	$M_k = \max_{0 \leq i \leq k} S_i \Rightarrow detection\, if\ M_k - S_k > \alpha$
$T_k = \sum_{t=0}^{k}(\zeta_t - m_0 - \frac{\delta_{min}}{2}), (k \geq 0)$	$N_k = \min_{0 \leq i \leq k} T_i \Rightarrow detection\, if\ T_k - N_k > \alpha$

Table 4.2: Équations du test de Hinkley

La moyenne m_0 est estimé à chaque instant. δ_{min} représente l'intensité minimal du saut qu'on veut détecter, et α est un seuil predefini. Dans nos expériences, nous prendrons α = 0.2 et δ_{min} = 0.2. L'idée principal du test de détection de sauts (vers le bas et vers le haut) est que S_k est une *integration* des déviations ζ_t par rapport à la moyenne m_0, qui dépasse un seuil $\delta_{min}/2$. Pour illustrer la méthode précédemment décrite, figure 4.2 montrent d'abord l'evolution de zeta sur une séquence vidéo de football de 10000 images et donne une idée de ce qu'on attend du test de Hinkley.

La prochaine section examine l'implémentation du second détecteur basé sur l'analyse d'histogrammes. Les résultats sont présentés et interprétés à la fin de ce chapitre.

Approche par analyse d'histogramme tridimensionnel RGB Cette deuxième implémentation est inspirée de [**?**]. Comme décrit dans l'état de l'art, les images consécutives dans un même plan vidéo contenant un aspect visuel similaire montrent de faibles différences dans leur histogramme comparé à des images de chaque côté de la frontière entre deux plans. La distance utilisée, connue pour ses performances pour comparer deux fonctions de densité de probabilité, est le test de χ^2:

$$z(k, k+1) = \sum_{j=0}^{n} \frac{|H_k(j) - H_{k+1}(j)|^2}{H_{k+1}(j)} \tag{4.13}$$

Figure 4.2 donnent une idée de la courbe de dissimilarité pour le même exemple de vidéo que la figure 4.2.

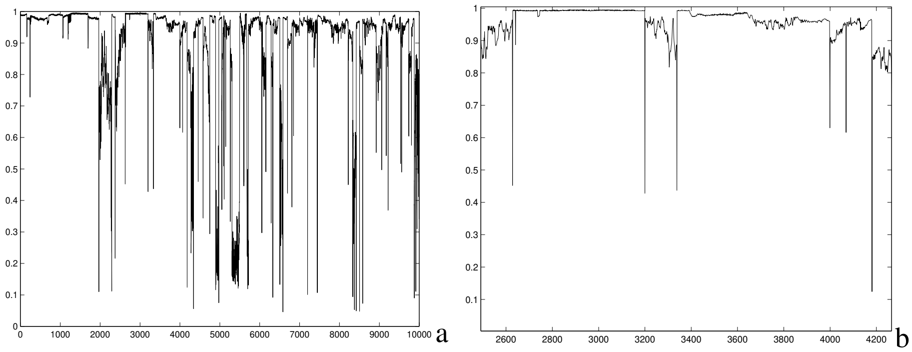


Figure 4.3: (a) Courbe ζ sur 6 minutes d'une vidéo de football (b) zoom sur la courbe ζ: les pseudo-diracs représentent les coupures entre deux plans, les segments où ζ est constant représentent des plans large shots sans beaucoup de mouvements à l'intérieur tandis que les segments bruités représentent des actions (zoom sur un joueur en action par exemple)

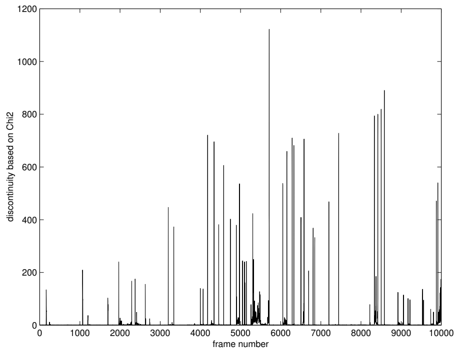


Figure 4.4: Courbe de discontinuité avec Chi2 - 6 min d'une vidéo de football

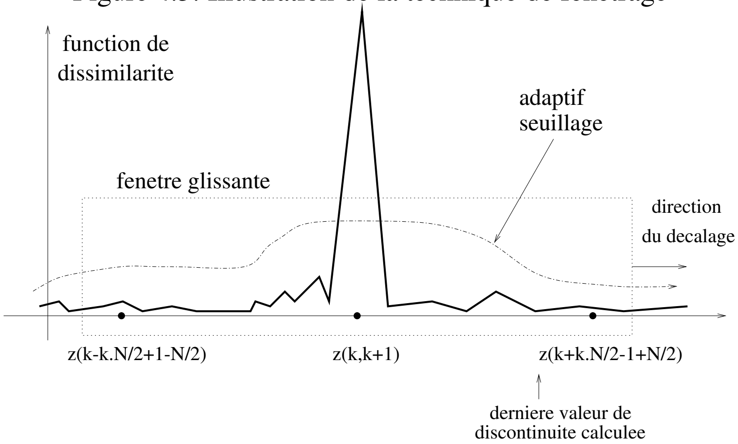
La méthode pour détecter les coupures dans la courbe z utilise un seuillage adaptatif T_k. Cette technique est moins sensible au "bruit" qu'un seuil fixe. Les valeurs $T(k)$ sont calculées à partir de l'information de la forme temporelle qui est caractéristique des coupures. Comme illustré figure 4.5, les N dernières valeurs consécutives de discontinuité sont considérées, formant une fenêtre gissante (4.5).

La présence d'un changement de plan est vérifiée à chaque nouvelle position de la fenêtre, à la valeur médiane de la fenêtre, suivant le critère de l'équation (4.14).

$$\begin{array}{l} si\, z(k,k+1) = \max_{i=-N/2,\ldots,N/2} z(k+i,k+1+i) \\ \wedge z(k,k+1) \geq \alpha z_{sm} \Rightarrow transition\, de\, type\, coupure \end{array} \tag{4.14}$$

En d'autres mots, une coupure est détectée entre les images k et $k+1$ si la valeur de discontinuité $z(k,k+1)$ est la valeur maximum de la fenêtre et α fois supérieure que la deuxième valeur de discontinuité z_{sm} à l'intérieur de la fenêtre (figure 4.5). Dans les expériences, nous avons fixé α = 6 et la longueur de la fenêtre $windowlength$ = 9.

Figure 4.5: Illustration de la technique de fenêtrage



Structure logicielle La figure 4.2 montre le modèle de classe utilisé. Le fait d'avoir codé ces deux algorithmes en C++ offre une meilleur réutilisation des programmes dans le futur, par exemple, si l'on veut fusionner ces algorithmes. Notons que dans le cas du test de Hinkley, on rentre en entrée de ce programme les données $zeta$. Ces données sont fournies grâce à un programme d'estimation de mouvement développé par Jean-Marc Odobez. Le programme d'extraction des coefficients χ^2 est relativement simple: chaque image est lue successivement et comparée en appliquant la formule (4.13) entre l'histogramme de image courante et celui de la précédente.

La classe *detection_Test* est composée **soit** d'une classe *Hinkley_Test* **soit** d'une classe *Sliding_Window_Test*. Le programme principal définit et initialise un objet de type *detection_Test* puis appelle la fonction *process()* qui gère elle-seul l'appel aux diférentes fonctions des différentes classes. Dans la classe *Sliding_Window_Test*, le type *CircArray* a été conçu pour extraire d'une façon efficace le maximum des parties à droite et à gauche de la valeur médiane dans la fenêtre glissante. La classe *Jump* identifie les sauts vers le bas et vers le haut.

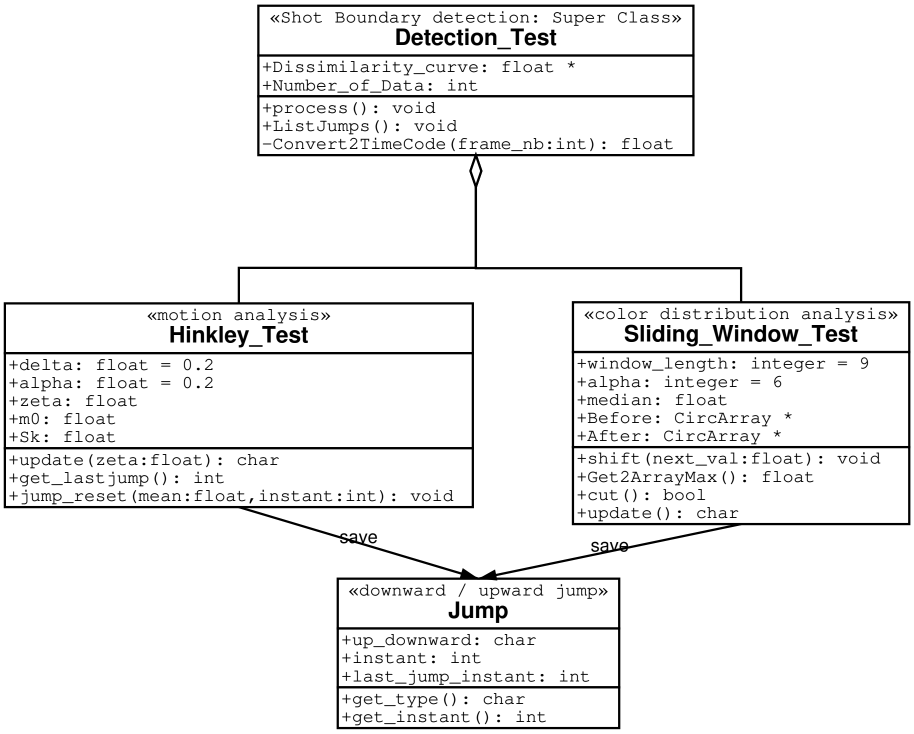


Figure 4.6: Modèle de classe représentant les principaux attributs et méthodes dans chaque classe.

4.3 Résultats de la détection des changements de plans

Deux transitions illustrent comment sont représentées une coupure et une transition graduelle suivant les deux courbes de similarité puis discutées. Une coupure (4.7) est représentée par un "dirac", inversé dans le cas de la courbe ζ, tandis qu'une transition graduelle est représentée par une "vague inversée". Notons que l'approche basé histogramme ne détecte pas les transitions graduelles comme l'illustre la figure 4.3.

- Transition de type coupure

  Les figures 4.7 montrent un exemple de type coupure. Une coupure est identifiée dans les deux approches par un -dirac- sur les courbes ζ (figure 4.7a comprise entre 0 et 1) et *dissimilarité* (4.7b comprise entre 0 et 1100) respectivement avec les analyse basé sur le mouvement et sur les histogrammes.

- Transition graduelle

  Une transition graduelle (figure 4.9) est identifiée differemment dans les deux approaches: L'analyse basé sur la distribution de couleur ne détecte pas ces transitions-ci. En effet, en comparant les histogrammes des images successives, la *distance* n'est jamais élevée visuellement. C'est le principal inconvénient de cette méthode. Figure 4.9a, la courbe χ^2 trace seulement un "dirac" représentant la coupure entre les deux dernier plan (figure 4.3c et 4.3d).

  En revanche, avec l'approche basée sur l'estimation de mouvement, une transition graduelle est représenté par une "vague" sur la courbe ζ comme montré figure 4.9b. Notons que dans le

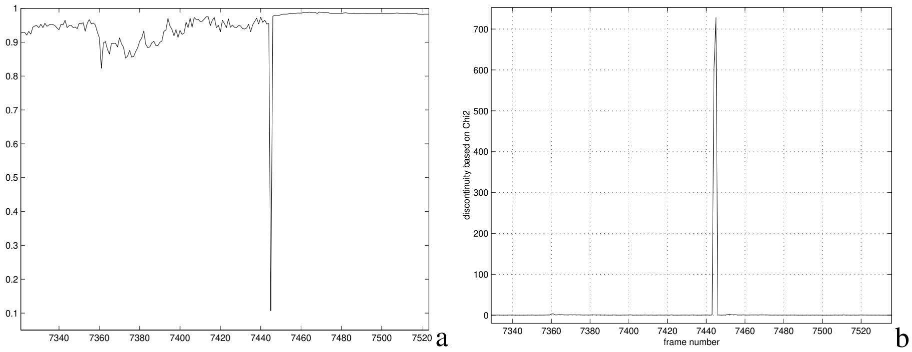


Figure 4.7: Example of a cut (a) zeta motion (b) histo Chi2

Figure 4.8: sample images from ASSAVID (a) (b) Before the cut frame numbers: 7361' 7444' respectively (c) (d) After the cut 7445' 7500' respectively

plan illustré par les images caractéristiques (4.9c et 4.9d), la courbe ζ "stationne" aux alentours de ζ = 0.2 (figures 4.9c et 4.9d). Ceci est interprêté par de trop importants mouvements dans ce plan. Une solution envisageable à ce problème serait de modéliser ce genre situation (par chaîne de Markov par exemple).

Ce dernier exemple montre bien les limites des deux algorithmes et du compromis auquel nous souhaitons converger. D'un côté, l'agorithme basé sur l'estimation du mouvement est sensible dans le cas où il y a trop de mouvement dans un plan (cf figures 4.9c et 4.9d). De l'autre l'algorithme basé sur la couleur ne détecte aucune transition. Il y a donc un compromis à faire et tout le problème réside dans l'extraction de descripteurs pertinents.

L'article [?] introduit un protocole standard d'évaluation de segmentation temporelle de vidéo. L'évaluation de la performance de ces deux algorithmes est inspirée de quelques formules tirées de cette étude. Les résultats des deux méthodes automatiques ont été comparés à la perception humaine (ou SVH, système visuel humain). Donc cette "vérité" est rentré sous forme de fichier donnant les numéros d'images exacts où se trouvent les changements de plans, plus particulièrement les sauts (vers le haut et/ou vers le bas) attendus. Ensuite, en ce qui concerne la méthode d'évaluation, la table 4.3 définit si un changement est bien détecté, manqué ou est une fausse alarme.

$prob\,(D = 1 \mid T = 1)$	$\Rightarrow$ Correct
$prob\,(D = 0 \mid T = 1)$	$\Rightarrow$ Missed
$prob\,(D = 1 \mid T = 0)$	$\Rightarrow$ FalsePositive or False Alarm
$prob\,(D = 0 \mid T = 0)$	$\Rightarrow$ Correct

Les deux formules de la table définissent les termes de précision et de rappel.

Recall = $\frac{Correct}{Correct+Missed}$	Precision = $\frac{Correct}{Correct+FalsePositive}$

Enfin les résultats sur le segment de 6 minutes d'un match de football sont présentés dans le tableau ci-dessous. Les résultats obtenus avec l'estimation du mouvement sont satisfaisants: moins de 10% de manqués. Ils sont légèrement meilleurs que ceux de l'analyse basée sur la distribution de couleur. Mais notons que cette deuxième méthode est nettement plus rapide. Une possibilité de fusion des deux algorithmes dans le futur est à envisager. Par ailleurs pour des résultats plus fiables, il faudra évaluer les méthode de détection de changements de plans sur, au moins, une vidéo de deux heures. Le Chapitre 5 analyse en détail les descripteurs pour la classification perceptuelle automatique de séquence vidéo.

	Analyse basée mouvement	Analyse basée histogramme
Nb plan vidéo	52	52
Correct	47	44
Missed	5	8
False Alarms	19	10
Recall	0.9 (mieux)	0.85
Precision	0.71(monis bien)	0.81

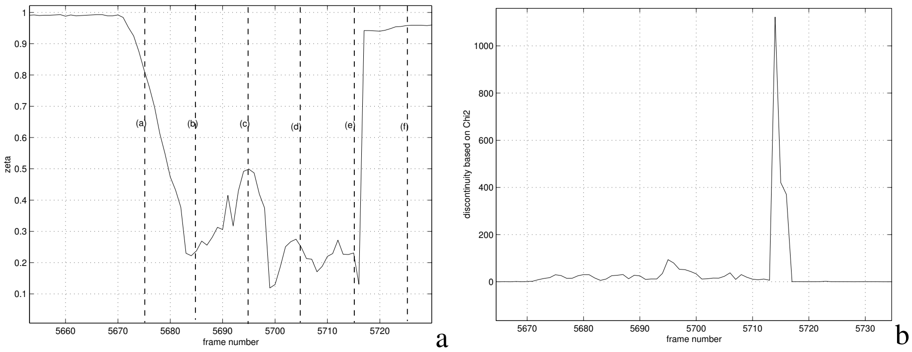


Figure 4.9: Example d'une transition progressive (a) courbe ζ liée à l'estimation de mouvement (b) courbe de dissimilarité χ^2, approche par histogramme

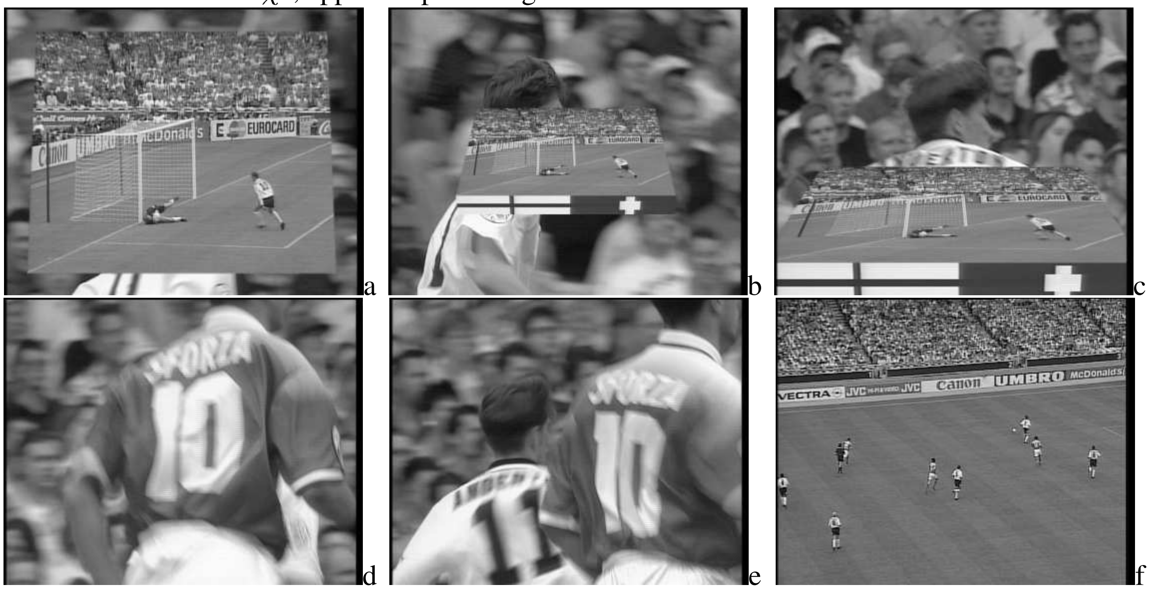


Figure 4.10: Images de la base de donnée ASSAVID correspondant aux courbes ci-dessus d'une transition progressive (a) début de la transition (b) (c) à l'intérieur de la transition: le dernier plan apparait dans la cassette vidéo (image de synthèse) avec deux drapeaux (jeux olympiques de Barcelone, Suède / Suisse) (c) (d) Plan très court sur un joueur (e) le prochain plan est un plan large

Chapter 5

Video shot features extraction

Where is and how to select the relevant information in an image and in a sequence of images ? Many types of features are extractable in a video sequence. Visual, motion, spatial (texture, object shape, etc.) and temporal information is usually exploited. Color distribution has proven its efficiency in content-based retrieval application ([**?**]). Also an efficient measure of similarity needs to be defined. This chapter deals with feature extraction and selection processes.

5.1 Key-frames selection

Three different Key-frame extractors are defined:

- Key-frames can be extracted randomly from a shot knowing the first and last frames.
- Key-frames can be regularly spaced knowing the first and last frames.
- We can detect the sub-shots boundaries starting from the first frame of a shot, we compare the next ones with that first one. Once the distance is up to a given threshold, a sub-shot is detected, a key-frame is extracted and we reiterate the algorithm.

This last technique is illustrated by the figure 5.1.

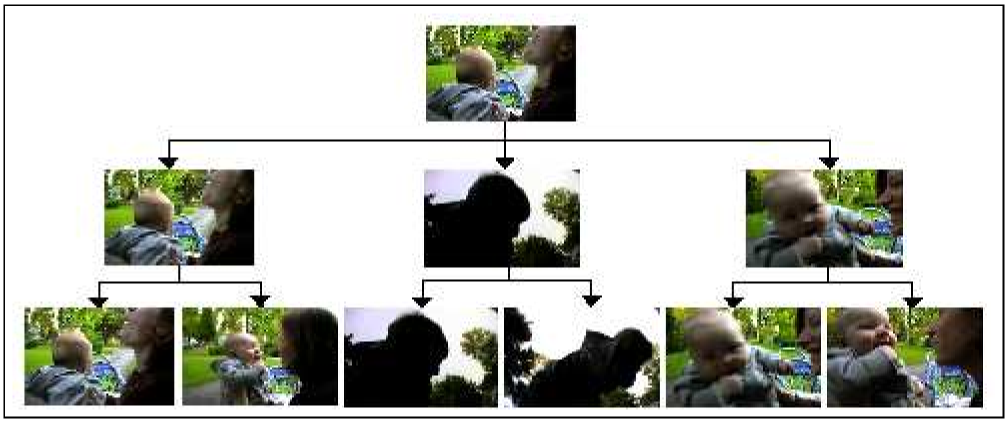


Figure 5.1: Shot hierarchy. The root node is the shot, the intermediate level is composed of sub-shots, and the leaves are random frames extracted from each shot.

5.2 Tri-dimensional RGB histogram-based analysis

One of the easiest techniques (but not the worst one) is to work with global color histograms. The use of multiple block histograms defining in regions of interest (ROI) might be interesting in special case only (for separating an image into three regions for instance). The main problems with the use of histogram is that it does not contain spatial information. On the other hand, we earn a bit using multiple block histograms but we loose if there is motion when comparing two frames. And in that case, it might be better to use global histogram. In our experiments, the 3-dimensional RGB histogram was selected as the visual information extracted from each shot key-frames. As already defined in Chapter 4, a normalized histogram is the Probability Distribution Function $H(j) = PDF\{I(x,y)\}$, where $I(x,y)$ is a gray level image or one RGB component of a color image and k is a gray value color component intensity on the abscissa axe. The intensity on the abscissa axe. The bhattcharrya coefficient, which coefficient, which has proven its interest in statistics has the property to be a metric[1] is computed between to 3-dimensional RGB histograms. The formal definition of the bhattacharrya metric where H is the PDF (normalized histogram in our case) is defined in equation 5.1:

$$z(k,k+1) = \sqrt{1-d(k,k+1)}\, where\, d(k,k+1) = \sum_{j=0}^{N-1} \sqrt{H_k(j) \times H_{k+1}(j)} \qquad (5.1)$$

Figure 5.2: Three images: (I_a) and (I_b) belong to the same scene while (I_c) belongs to another one

I_a I_b I_c

From the three images presented figure 5.2, table 5.2 gives the Bhattacharrya coefficient associated with the PDF dissimilarity computation.

Table 5.1: Bhattacharrya metric examples

$d_{BT}(PDF(I_a), PDF(I_b))$	0.197003
$d_{BT}(PDF(I_a), PDF(I_c))$	0.756017

We can note from figure **??** that each R-G-B component has its own information. Using the whole data gives more accurate information compared to gray-scale histogram.

[1]The metric axioms : (1) Distances are non-negative. (2) The distance between two sequences can be zero only if the sequences are identical. (3) The distance from sequence A to sequence B is equal to the distance from sequence B to sequence A. (4) The distance between sequences A and B is smaller than (or equal to) the sum of the distances A->C and B->C. (the triangle inequality)

5.3 Temporal distance computation

In home videos, clusters are *localized* in time, and therefore strong connectivity conditions can be applied for clustering, which has the benefits of computational simplicity.

The temporal distance is normalized following this equation (figure 5.3 illustrates the method):

$$d_T(i,j) = \frac{|d_i - d_j|}{d_{total}} \tag{5.2}$$

where d_{total} represents the total duration of a given video sequence. Home video clusters composed of non-adjacent shots (forward and backward temporal jumps) are infrequent. In our database, only about 3% of the clusters present this characteristic.

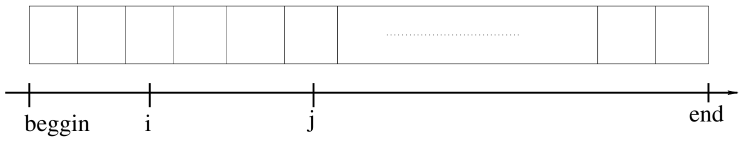


Figure 5.3: A video is a sequence of images ...

In this chapter, we have suggested the development of methods that

1. integrate segment visual similarity and duration in a joint model,
2. rely on strong temporal adjacency, and
3. account for the fact that clusters are likely to contain only a few elements.

Chapter 6

Video shot clustering

Video shot clustering is the heart of this project as a real contribution to research in this area. This chapter presents at first typical methods for clustering video shots and introduce spectral clustering. The second section examined in detail one recent algorithm and last section gives our clustering results on home video database.

6.1 State of the art

General concepts Clustering is a powerful technique used in various disciplines such as speech analysis, biotechnology, information retrieval, economy, etc. and especially in machine learning and pattern recognition. General methods are briefly reviewed here [**?**]: K-means finds randomly K centroids in a given dataset and iterates until the algorithm has converged. This is a simple, effective procedure but performs poorly with overlapping clusters. Gaussian mixture models are probabilistic ("soft") versions of K-means. Each mean (centroid) is view now as means of Gaussian distributions. The EM algorithm is a method for maximum likelihood (ML) parameter estimation. By giving clustering a probabilistic foundation, "clustering distance function" is replaced by "likelihood under a model". Hierarchical agglomerative clustering (HAC) allows to hierarchically define clusters from a set of points.

Before going deeply through the spectral method, previous techniques and typical approaches for video structuring are discussed. Then a paragraph present briefly last publication on spectral methods for machine learning.

Typical approaches for video structuring Cluster analysis constitutes one of the challenges of video analysis. In particular, hierarchical agglomerative clustering (HAC) methods have been used in the past. Early work proposed versions of either purely visual or time-constrained shot clustering, without specifically addressing home video. Specifically, the work in [**?**] proposed the use of a time-constrained full-link HAC, which required the heuristic set up of a number of thresholds for visual and temporal features, and the used of a transition graph for browsing. The methodology was applied for structuring of TV programs and movies.

- The method proposed by Yeung et. al. [**?**] is based on the concepts of time constrained clustering of video shots and on the construction of a scene transition graph to automatically parse a video program, extract story structures and identify story units. The authors define a Scene Transition Graph (STG) to decompose a video hierarchically (act-scene-shot). They derive from the Scene Transition Graph a proximity matrix having as entries the dissimilarity between two shots. The dissimilarities of the newly merged clusters to other clusters are updated at each step of their algorithm.

The proposed framework for the video analysis can be decomposed into the following functional blocks:

- compressed video: the video sequence is temporally subampled
- shot segmentation
- time-constrained clustering of video shots
- building of the Scene Transition Graph
- scene segmentation > to reduced number of data

They propose video shots to be the unit of organization. In order to segregate one scene from another, they take into account temporal locality of contents in video. This means that for any two shots that are far apart in time, even if they share similar visual contents, they potentially represent different contents or occur at different scenes. They impose a time-window parameter T that prevents two shots that are far apart in time but similar to be clustered together. In a STG, a cut edge connects two disjoint connected subgraphs, each of which is a story unit. They thus use a story unit to approximate a scene for the segmentation purpose. The scene transition graph helps them to identify where the scene cuts are.

- Hanjalic et al. [**?**] mainly focus on finding an alternative way of manually abstracting a video by attempting to map human cognition onto the machine level. Their abstraction method consist of three major phases:

  - partitioning clusters to all frames of a video sequence,
  - automatically finding the optimal combination(s) of clusters by applying the cluster-validity analysis and finally
  - representing each of the clusters by one characteristic frame (namely key-frame. The preview sequence is therefore obtained by concatenating all video shots to which the extracted key-frames belong.

  This method improves the way to extract key-frames but has a main drawback: temporal information is not considered. Also, abstraction is not an easy task to evaluate.

- To situate this study in the framework proposed by Rui et al.[**?**], we concentrate on the first two areas of research which are: video analysis and video representation. The two other fields, more high level are video browsing and retrieval. A hierarchical video representation is derived from key-frames (spatial feature), shots (temporal feature), groups, scenes to video. In the proposed framework, a group is a simple grouping of shots while a scene is a construction of a video representation at the semantic level. The authors suggest different ways to extend spatial information: to capture the appearance of the entire background present in the shot (a few mosaic images) and to extend temporal and geometric information introducing motion of camera or/and moving objects scene, group and shot similarities. Hierarchical video representation

Video analysis is defined using several representations: sequential key-frame representation, group-scene based representation and video mosaic representation.

- d/ Gatica et al.[**?**] do not use any graph analysis. The authors investigate visual and temporal features of home video and learn probability models that are used for clustering based on an optimal criterion. This probabilistic formulation avoids the use of heuristics that are hard to define, and allows to model multiple features in a unified way (joint distribution), and to introduce additional knowledge using prior distribution.

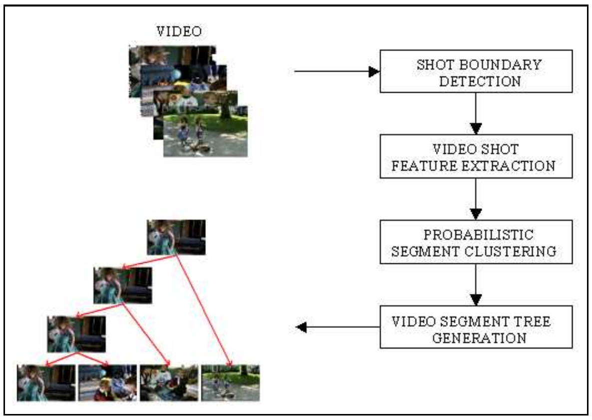


Figure 6.1: Architecture for Consumer Video Structuring in Gatica et al. [**?**].

Gatica et al. [**?**] shares the Bayesian methodology with a number of recent approaches (shot boundary detection, and still image classification). In their case, the authors want to disclose the cluster structure of videos for which the number of classes and their meaning cannot be predefined. A pre-established number of classes could be used for clustering, but would be quite limited for unrestricted content. Specifically to video structuring, their work is related to the work in [**?**], but it is distinct in several ways. Unlike [**?**], their work systematically investigates visual and temporal features of a specific source of video, and learns probability models that are used for clustering based on an optimality criterion.

A key-result in the work [**?**] is the demonstration that it really exists a structure in consumer videos. This chapter is a continuation of this work.

The main problem of the first three reviewed methods is that a lot of heuristic parameters and thresholds need to be tuned. On the other hand, the probabilistic method proposed by Gatica et al. [**?**] needs a lot of training data.

Recently, a number of authors have suggested alternative segmentation methods that are based on eigenvectors. The study presented in kannan, vempala and vetta [**?**] provide a theoretical analysis of spectral clustering defining a measure of the quality of a clustering. The authors modelise the clustering problem via an edge-weighted complete graph whose vertices need to be partitioned. The weight of an edge a_{ij} represents the similarity of the vertices (points) i and j. A cluster will be a subgraph. Weiss [**?**] uses properties of block matrices to analyze spectral algorithms. He attempts to prove results on idealized block matrices and then appeal to perturbation theorems on eigenvectors. Four spectral algorithms are reviewed: The similarities are that they all use the top eigenvectors of a matrix. They differ in two ways: which eigenvectors to look for and whether to normalize the A

matrix in advance. Meila and shi [?] suggest to view the pairwise similarities as edge flows in a Markov random walk and study properties of eigenvectors and eigenvalues of the affinity matrix.

An intuitive introduction to spectral clustering Spectral clustering has recently interested researchers in both fundamental (machine learning, pattern recognition) and applied research areas (image segmentation). This method exploits concepts from linear algebra ([?]): It refers to the general technique of partitioning the rows of a matrix according to their components in the top few singular vectors of the matrix. In contrast to statistical clustering methods that assume a probabilistic model that generates the observed data points, pairwise clustering defines a similarity function. Recently, in pattern recognition and computer vision research areas, a number of authors [?, ?, ?] have suggested alternative segmentation methods that are based on eigenvectors of the "affinity matrix".

Figure 6.1a shows three clusters of 2D synthetic points and figure 6.1b shows the affinity matrix defined by:

$$A(i,j) = e^{-d_{(x_i,x_j)}/2\sigma^2} \tag{6.1}$$

with a free parameter σ. In this case, we have used $d_{(x_i,x_j)} = ||x_i - x_j||^2$. The A matrix can be seen as encoding the pairwise similarities of vertices of a graph. Note that we have ordered the points so that all the points belonging to the first cluster appear first and then the one from the second cluster. This helps the visualization of the matrices but does not change the algorithms: eigenvectors of permuted matrix are eigenvectors of the original matrix.

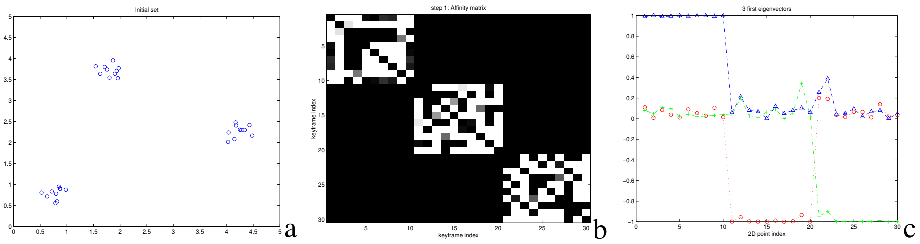


Figure 6.2: (a) A simple clustering problem (b) The affinity matrix (c) 3 first eigenvectors (Rows of Y in the algorithm)

Let's have a look at figures 6.1 and more in details in figure 6.1c. Automatic perceptual segmentation is based on the Human Visual System (HVS). And one of the goals of computer vision (including video analysis) is to try to achieve comparable performance. One efficient way early discovered, as mentioned earlier, is the use of eigenvectors. When a human looks at three clouds of points, what naturally retains his attention is the fact that points are grouped into three clusters. The machine cannot achieve a comparable task without a machine learning algorithm, called clustering algorithm. The plot in figure 6.1c represents the three first eigenvectors of the rearranged affinity matrix. These three eigenvectors gives us crucial information. This is all the heart of this method: it is much easier to cluster in this form rather than the original one. This transformation allows to represent the data in a new space much easier to cluster. In effect, the plots can be partitioned into three parts. Three patterns appear when the three curves are taken into account. Figure 6.2 gives another even clearer representation plotting 3-dimensional points, but before explaining the result, next section examines the used algorithm.

6.2 Ng et al. algorithm for spectral clustering

Spectral algorithms have been recently used for image segmentation problems (see [**?**]) but never for video segmentation (as far as we know). Video shot clustering is also an application of perceptual grouping algorithms. A contribution to actual research in the field of Video Analysis as a novel application of a recent algorithm is presented in the two following sections. Particularly, Ng et al. algorithm [**?**] shows surprisingly good results on 2D synthetic points examples. We illustrate these results as well as the application of this algorithm in the last section.

Given a set of points $S = s_1, ..., s_n$ in R_l that we want to cluster in k subsets:

1. Form the affinity matrix $A \in R^{n\times n}$ defined by A_{ij} = exp(-$||s_i - s_j||^2/2\sigma^2$) if $i \neq j$, and A_{ii} = 0.
2. Define D to be the diagonal matrix whose (i,i)-element is the sum of A's i-th row, and construct the matrix $L = D^{-1/2}AD^{-1/2}$.
3. Find $x_1x_2...x_k] \in R^{n\times k}$ by stacking the eigenvectors in columns.
4. Form the matrix Y from X by renormalizing each of X's rows to have unit length (i.e. $Y_{ij} = X_{ij}/(\sum_j X_{ij}^2)^{1/2}$).
5. Treating each row as a point in R^k, cluster them into k clusters via K-means or any other algorithm (that attempts to minimize distortion).
6. Finally, assign the original points s_i to cluster j if and only if row i of the matrix Y was assigned to cluster j.

Table 6.1: Ng et al. [**?**] spectral clustering algorithm

How does this algorithm work? As already discussed, we know from the previous section why do we build the affinity matrix. We also have an intuition about the use of eigenvectors from figure 6.1. Let's following each step from Ng et al. algorithm presented in table 6.2. To understand the algorithm, it is instructive to consider its behavior in the "ideal case" like it is almost the case in the example of figure 6.1a. Roughly speaking, at step 2, the L matrix is build from A so as to eliminate the "noise" around the blocks and, in this way, try to approach the ideal case of a perfect block diagonal-like matrix. Note that $\hat{A}$ and $\hat{L}$ are block diagonal.

$$\hat{A} = \begin{bmatrix} A^{(11)} & 0 & 0 \\ 0 & A^{(22)} & 0 \\ 0 & 0 & A^{(33)} \end{bmatrix}, \hat{L} = \begin{bmatrix} L^{(11)} & 0 & 0 \\ 0 & L^{(22)} & 0 \\ 0 & 0 & L^{(33)} \end{bmatrix}. \tag{6.2}$$

As an explanation of step 3, eigenvectors of a block diagonal matrix are ideally one for the index belonging to one block (cluster) and filled with zeros otherwise. To construct $\hat{X}$, we find $\hat{L}$'s first k=3 eigenvectors. Since $\hat{L}$ is block diagonal, its eigenvalues and eigenvectors are the union of the eigenvalues and eigenvectors of its blocks. A basic result from matrix theory ([**?**]) is that, with diagonal matrices, the first eigenvalue = 1. Thus, stacking L's eigenvectors in columns to obtain X, we have:

$$\hat{X} = \begin{bmatrix} x_1^{(1)} & \vec{0} & \vec{0} \\ \vec{0} & x_1^{(2)} & \vec{0} \\ \vec{0} & \vec{0} & x_1^{(3)} \end{bmatrix} \in R^{n \times 3}. \tag{6.3}$$

The three first eigenvectors of $\hat{L}$, the three rows of X here, are orthogonal. The inner product of any of the eigenvectors is 0: $< V_i.V_j >= 0$. Next, by renormalizing each row of $\hat{X}$'s rows to have unit length, we obtain:

$$\hat{Y} = \begin{bmatrix} \hat{Y}^{(}1) \\ \hat{Y}^{(}2) \\ \hat{Y}^{(}3) \end{bmatrix} = \begin{bmatrix} \vec{1} & \vec{0} & \vec{0} \\ \vec{0} & \vec{1} & \vec{0} \\ \vec{0} & \vec{0} & \vec{1} \end{bmatrix} \in R^3. \tag{6.4}$$

where $\hat{Y}^{(}i) \in R^{n_i \times k}$ denotes the i-th sub-block of $\hat{Y}$. Step 3 and 4 are crucial. As previously discussed, all the information resides in the top eigenvectors. In step 3, by staking these K eigenvectors in column, we get n points in R^K. In step 4, the fact to normalize the rows brings all the points at the surface of a K-sphere. The authors end up the algorithm description concluding that there are k mutually orthogonal points on the surface of the unit k-sphere around which $\hat{Y}$'s rows will cluster. Moreover, these clusters correspond exactly to the true clustering of the original data. To switch in the general case, matrix perturbation theory [**?**] indicates that the stability of the eigenvectors of a matrix is determined by the *eigengap*. More precisely, the *subspace* spanned by $\hat{L}$'s first eigenvectors will be stable to small changes to $\hat{L}$ if and only if the eigengap $\delta = |\lambda_3 - \lambda_4|$, the difference between the 3rd and the 4th eigenvalues of $\hat{L}$, is large (we suppose here that K = 3). From this discussion around a spectral algorithm, we can interpret that eigenvectors are interesting when clustering a set of points in R^n.

Eigenvalues and eigengap The following two figures illustrate the computation of the eigengap from the eigenvalues. These examples are extracted from Rylan 0 video where we expect nine scenes, that is K = 9 clusters.

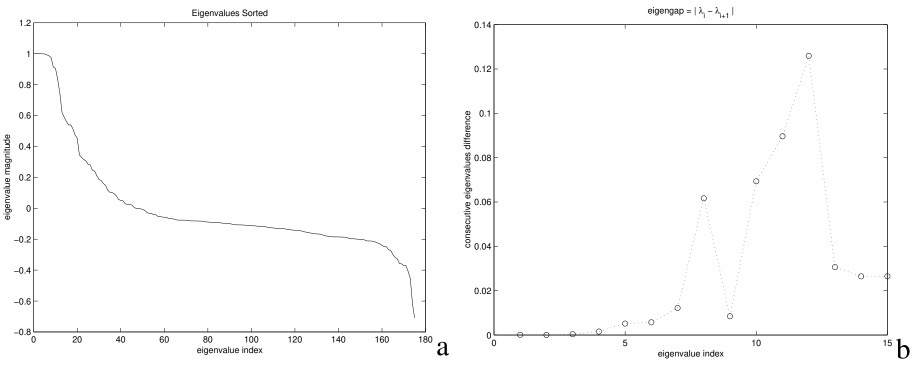


Figure 6.3: from Rylan 0 video (a) sorted eigenvalues and (b) eigengap

Generation of 2-D synthetic examples On the previous data set of three well separated clusters, a simple K-means manage perfectly the partition. Now, let's have a look to two other challenging 2D

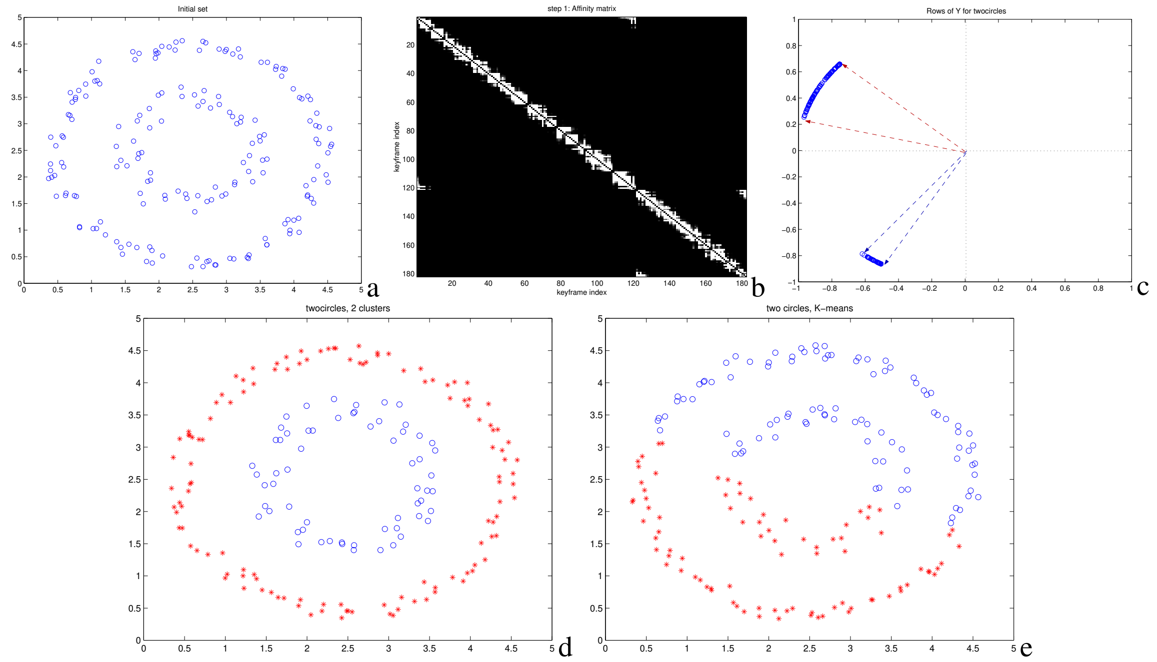


Figure 6.4: Clustering example (a) original (b) Affinity matrix (c) Rows of Y (d) Spectral clustering result (e) results with K-means

synthetic examples as an illustration of the results that can be obtained with this method. A σ of 0.013 has been used here.

In this first synthetic example (figure 6.2), the dataset consists of points representing two circles. With our HVS, this is instantaneously that we conclude that there is two clusters: the small one and the large one. Once again, the machine does not know that because of a lack of learning knowledge. In this particular example where K-means procedure fails dramatically (Figure 6.2e), how does the spectral clustering technique performs ? Figure 6.2b represents the affinity matrix. Note that the two clusters are far to be clearly distinguishable. Only we can observe four small white spots on each side of the matrix: this occurs when the Euclidean distances finishes to go through the circles. Figure 6.2c shows the rows of Y. The two clusters are now nicely distinguishable by performing K-means. The result of the spectral clustering is displayed Figure 6.2e.

The figures displayed in 6.2 refers to the same functions than the previous ones but with a different dataset of three clusters 6.2. The two first block of the affinity matrix 6.2b corresponds to the two clouds while the left-bottom corner represent Euclidean distance between points on the curved line. Again, applying directly K-means gives bad clustering results 6.2e. The rows of Y (here jitter ed) are far apart in terms of angle: the centroids are orthogonal. It is therefore easy now for K-means to find the good clustering. Note that “good clustering” means human perceptively speaking.

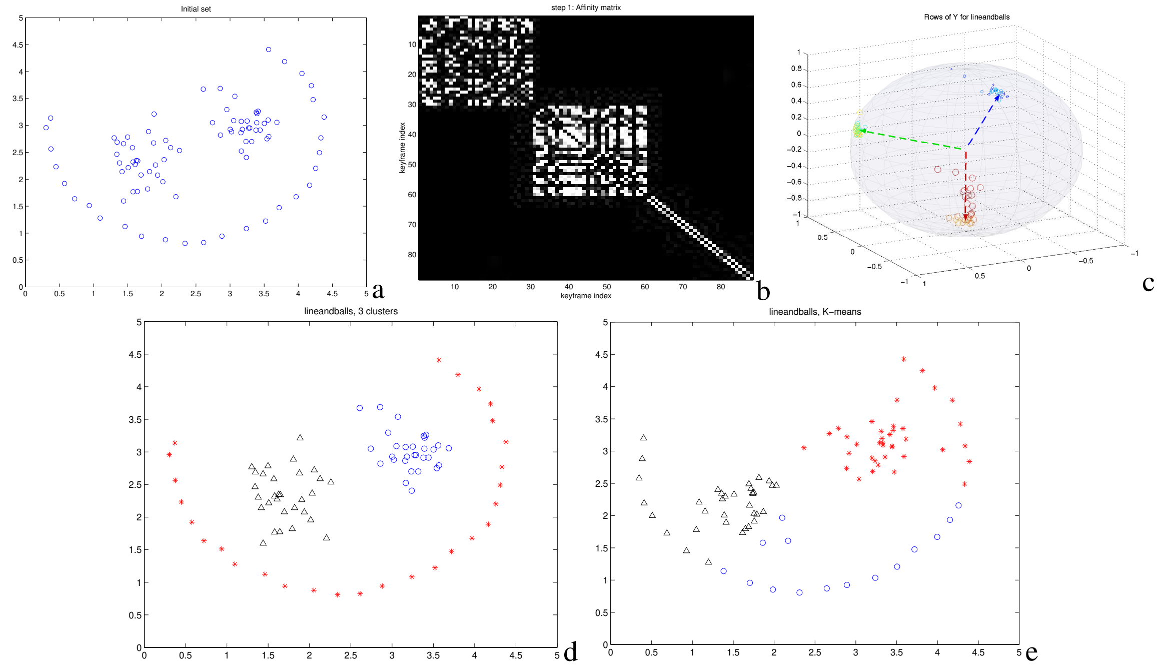


Figure 6.5: Clustering example 3 clusters (a) original (b) Affinity matrix (c) Rows of Y (d) Spectral Clustering result (e) results with K-means

As noticed in the discussion of the algorithm section 6.2, matrix perturbation theory ([**?**]) informs us about the possibility to guess automatically the number of cluster from the eigengap computation. If we take a look at figures 6.2, we can see that for the *twocircles* example, the two first eigen-value are more or less closed to 1 ($\lambda(0) = 0$) and the other one different. Also, in the *lineandballs* challenging dataset example, the three first eigen-values are closed to 1 ($\lambda(0) = 0$) while the next one is different. What we can observe from these two following plots is that the eigengap, for the abscissa value K is higher than the previous values. So a first solution to the problem of guessing the number of clusters would be to perform this signal processing test. But we will see in further discussions that it is not as clearly remarkable in eigengap plots derived from video sequences.

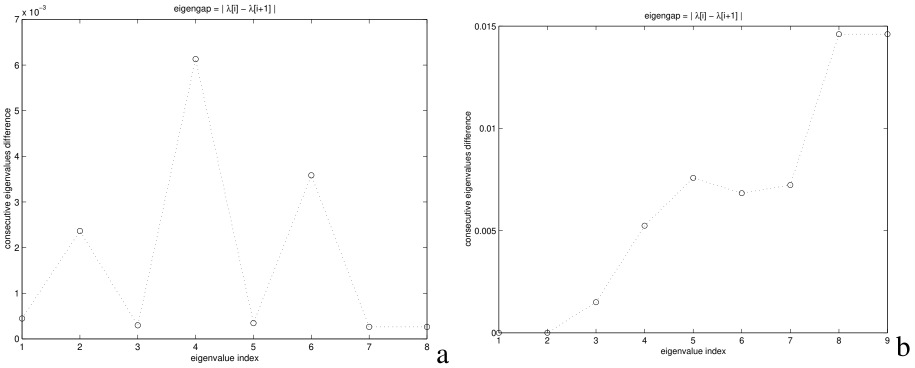


Figure 6.6: Eigengap plots for (a) twocircles example and (b) lineandballs example

6.3 Application on video structuring

6.3.1 Experimental setup

a/ Software design

Globally, the software is divided into three parts.

1. A software coded in C extracts two matrices from a video:
   - d_{BT} (bhattacharrya)
   - d_T (temporal)
2. The spectral clustering algorithm described in 6.2 is implemented under Matlab.
   - Reads the two matrices
   - Processes spectral clustering algorithm
   - Generates the clustering masks
3. A C program from Dr. Daniel Gatica-Perez outputs the performance evaluation results given the ground truth file.

b/ Selection of the distance function

In this part, we try to find the optimal distance combining the visual similarity metric with the temporal distance. d_{BT} and $d_T$refer respectively to Batthacharrya metric and temporal distance defined section 5.

We present here three distance functions that compare key-frames:

- Euclidean distance:

$$d^2 = d_{BT}^2 + d_T^2 \tag{6.5}$$

- Weighted sum (L1 distance):

$$d = \alpha_1.d_{BT} + \alpha_2.d_T \tag{6.6}$$

- Non- linear distance that includes cross-products:

$$d = \log_2 (d_{BT} + 1)(d_T + 1) \tag{6.7}$$

The only way to compare them is heuristically: we found that the best results are achieved taking the weighted sum as the distance function.

c/ Model selection

The “best” value of K is not always unique. Experiments was run over the whole database for different values of K (including the “ideal”K as reported in the database), and the “best” results that are obtained are shown in section 6.3.2.

d/ Initialization of the K-means procedure

Ng et al. algorithm [**?**] is sensitive to initialization: if no initialization of the K-means procedure is processed, then the centroids are randomly assigned which results in a poor grouping. In order to handle this two different initializations have been carried out The goal is to select K rows of the Y matrix that are orthogonal in order to get k orthogonal centroids before starting K-means procedure. Two coded algorithms perform this task:

1. A first initialization of the K-means procedure has been performed inspired from Gram-Schmitt orthogonalization. For each newly selected centroid W, we calculate $W_{\perp}$ which is strictly orthogonal to others already picked up. The key of the computation is that two vectors V_1 and V_2 are orthogonal if inner product $< V_1.V_2 >= \vec{0}$.

2. A simplified orthogonalization was also coded:

   - Let the first centroid be randomly chosen row of Y,
   - Repeatedly, choose the next centroid the row of Y that is closest to being 90 degrees from all the centroids (formally, the worst-case centroid) already picked.

The two initialization differ only on the following point: the first one build strictly orthogonal centroids from the best matching one while the second just pick the “best” matching row as a new centroid.

6.3.2 Experience and results

Description of the kodak home video (c) database The data set consists of 30 MPEG-1 video clips of different characteristics. Each sequence has a duration between 18 and 25 minutes, and was digitized from VHS tapes at 1.5Mb/s in SIF format. The total duration is nearly ten hours (about 1075000 frames). The data were collected from eleven different people, and is representative of consumer video content: indoor and outdoor scenarios, depicting weddings, vacations, children playing, school parties, etc. A third-party ground-truth at both the shot and the cluster levels was manually generated (see Section 6.3.2 for further discussion). Additionally, garbage video shots (transitional shots with no content), and very poor quality shots were not taken into account for the experiments. These two situations mimic human handling of poor quality still pictures. After this adjustment, the total number of shots and clusters in the database are 801 and 189, respectively. Both the number of shots and the number of clusters per sequence vary considerably (between 4 and 105 shots, and between 1 and 19 clusters, respectively). As far as we are aware, this constitutes the most extensive home video database reported in the current literature

In the used database, a third-party cluster ground-truth was determined based on human evaluation of shot visual similarity, temporal adjacency, and *blind* context understanding.

Intermediate results from the spectral algorithm After having analyzed the Ng et al. algorithm, the following two figures present intermediate results on Rylan 0 video. This video is composed of 9 scenes. We can distinguish them by counting the number of blocks in the affinity matrix. The eigengap figure highlights the idea of automatically guessing the number of clusters by either detecting the first Dirac on the curve or applying a more robust 1D processing. A possibility would be, like in the Hinkley test, to count successively the area under the curve (an simple *integration*) and threshold it.

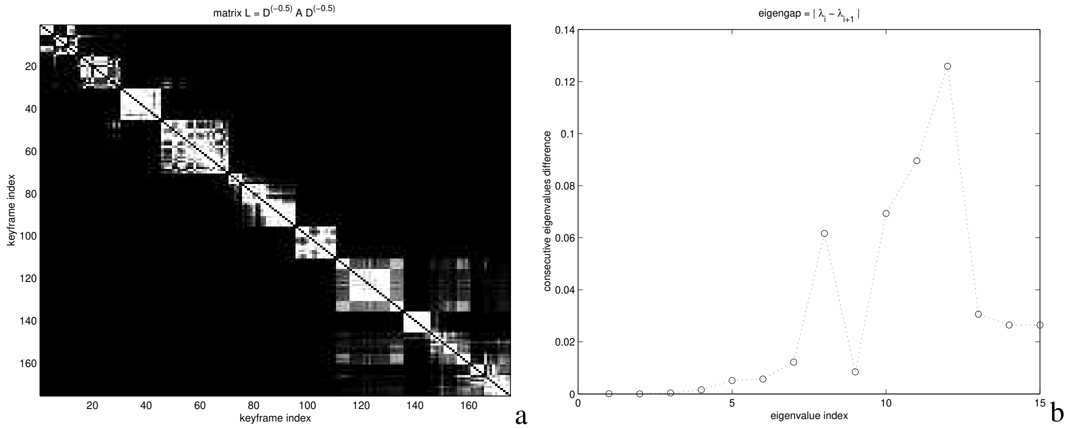


Figure 6.7: Clustering “RYLAN 0” video (a) Affinity matrix (b) Eigengap

Performance Evaluation Procedure This section defined the different variables. Note that in our experiments, the number of cluster is chosen ideally because an algorithm which guess automatically the number of clusters has not been implemented yet. In section 6.4, we discuss the model selection relatively to this point.

Given NC, the number of clusters in the ground-truth (either in an individual sequence or in the whole database), the criterion $\mathcal{C}_1$ is evaluated by defining three variables: *Detected Clusters* (DC), which indicates the number of clusters that were found by the algorithm, *False Positives* (FP), defined by $FP = DC - NC$ if $DC - NC \geq 0$ and zero otherwise, and *False Negatives* (FN), defined by $FN = NC - DC$ if $DC - NC \leq 0$ and zero otherwise. To evaluate the criterion $\mathcal{C}_2$, *Shots In Error* (SIE) is used to denote the number of shots whose cluster label does not match the label in the ground-truth. Finally, *Correcting Operations* (CO) indicates the number of operations (merging/splitting) needed to correct the results so that SIE is zero. We believe this is a good indication of the effort required in interactive systems. We are interested in analyzing the performance figures as probabilities. If z is any of the parameters of interest, the frequentist evaluation of the performance produces two typical estimates: the *macro-average* z_M, which is directly computed over the whole database, and the *micro-average* z_m, in which the figure is first estimated for each individual sequence, and then averaged over the whole database. While the former assigns the same importance to each shot (or cluster) in the database, the latter gives the same importance to each video sequence, regardless of the number of shots or clusters it contains. We present both figures for discussion. For macro-averages, if $FP = 0$, the figures are computed by

$$dc = \frac{DC}{NC}; \quad fp = 0; \quad fn = \frac{FN}{NC}, \tag{6.8}$$

and if $FN = 0$, the expressions are

$$dc = \frac{DC - FP}{DC}; \quad fp = \frac{FP}{DC}; \quad fn = 0. \tag{6.9}$$

Additionally,

$$sie = \frac{SIE}{NS}; \quad co = \frac{CO}{NS}, \tag{6.10}$$

where NS stands for the number of shots. For micro-averages, eqs. 6.8 6.9 and 6.10 are valid for each individual sequence. Results are then accumulated and averaged over the whole database.

Table

Shots	8	NumberOfGroups	3
DC	3	number of clusters	3
FP	0	SIE	0 1 0 0 0 0 0 0
FN	0	SIE2	0 1 0 0 0 0 0 0
SIE	1	GT	0 1 1 1 1 1 2 2
CO	1	Clusters	0 0 1 1 1 1 2 2
List of SIE	1	probing clusters of size 1	shot 0

Table 6.2: Performance evaluation output, An example

In order to compare the spectral clustering method examined in this chapter with other known methods, let define two simple ways in table 4.1.

In the table 6.3.2, six methods are compared. B_0 and B_1 are definedas :

- B_0 : segmentation where scenes boundary are regularly temporally spaced, assigns a uniform and temporally adjacent number of shot per cluster.

- B_1 Kmeans clustering for shots, in which the centroids were initialized with randomly selected shots from each sequence.

Probabilistic Clustering refers to the method used in Gatica et al. work [**?**]. Then The three at the bottom of the table are related with the spectral method tuning three heuristcics:

- K: the number of clusters,
- σ: the free parameter from the affinity function computation,
- the distance function: $d = \alpha_1.d_{BT} + \alpha_2.d_T$,

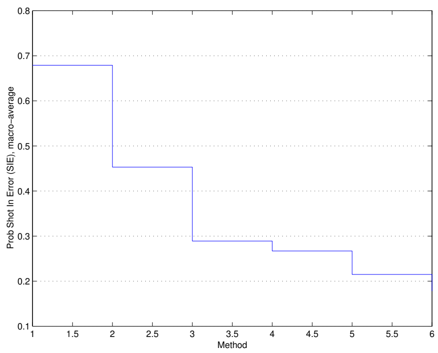
The first one displays results inputing the good number of clusters K. The second displays the results where the "best" σ (from 0.08 to 0.2) in each video were found. And the last line refers to selectioning the best results varying σ and K (+/- 2 from the ground truth segmentation). Line 4 of the table 6.3.2 refers to results with $\sigma = 0.0875$ and the following weighted sum:

$$d = 0.45.d_{BT} + 0.55.d_T \tag{6.11}$$

Method	sie_M	co_M	sie_m	co_m
B_0	0.679	0.609	0.588	0.529
B_1	0.453	0.167	0.430	0.200
Probabilistic Clustering	0.289	0.133	0.286	0.173
Spectral Clustering (fixed σ)	0.267	0.158	0.234	0.149
Spectral Clustering (best σs)	0.215	0.143	0.178	0.130
Spectral Clustering (best σs & Ks)	0.178	0.119	0.144	0.109

Table 6.3: Shot Assignment Performance

Figure 6.3.2 displays in a more visualable form the variable sie_M describing the percentage of shots in error from table 6.3.2.



a

Figure 6.8: (1)Regularly spaced segmentation (2)K-means only (3)Probabilistic Clustering (4)Spectral Clustering (fixed σ) (5)Spectral Clustering (best σs) (6)Spectral Clustering (best σs & Ks)

6.4 Discussion and limitations

This method showing surprisingly good results on 2D synthetic challenging examples is also very interesting for video shot clustering. This technique does not need to train data: we are not estimating any models. The number of computations is very low compared to typical methods: The algorithm was implemented under Matlab and the results are almost instantaneous. The figure 6.4 displays as a browser the 135 key-frames clustered of RYLAN 0 video. It is important to look at carefully the table of results displayed in 6.3.2. The last line displays the "best" results we got varying three parameters and running them over the entire database of 30 videos. Results are detailed in 6.4. Here are the parameters we tuned:

1. distance parameters: L1 distance with different α: $0.45 < \alpha_1 \& \alpha_2 < 0.55$, product distance, Logarithm of the sum,
2. the number of cluster: K = -2, -1, 0, +1, +2, from the expected ideal case
3. σ: from 0.08 to 0.25

The "ideal" combination of these parameters are different from one video to another. Nevertheless, the distance which comes back very often as optimal is the L1 distance. And depending on the structure of a video, the range of the best σ is from 0.085 to 0.2.

Limitations

- An adaptative σ technique has still not been implemented.
- Figure 6.4 shows some examples of shots assigned to a wrong cluster.

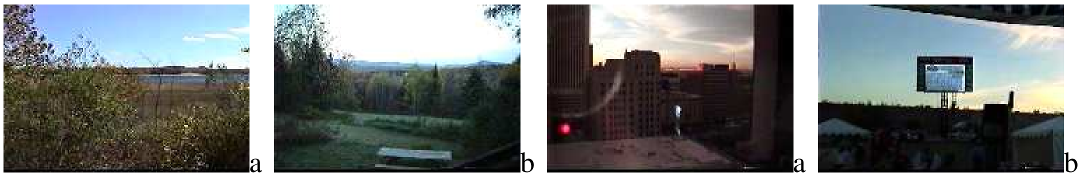

a b a b

Figure 6.9: limitation in shot assigned to a wrong cluster

- When testing the spectral algorithm with 2D synthetic examples, the more we have points inside each set, the easier it is to find the clusters. So a minimum of key-frame need to be extracted.

  For these experiences, keyframes have been extracted regularly spaced. Another method would be to detect sub-shots by comparing each keyframes from the first and deciding that there is a new keyframe when the distance between the two are above a threshold τ. Another possibility is to pick them randomly inside the shot as described in the previous chapter.

- Ground truth Clustering is not unique. Human evaluation can be seen as a probability distribution. These data will be taken into account in the future in order to give the right evaluation to a given algorithm.

- The problem of guessing the number of clusters within a video is still open using eigen-gap. For the challenging 2D synthetic examples as well as for videos where the structure is clear, eigen-gap can be relevant to guess the number of cluster. But in difficult cases, the eigengap does not reflect very much the expected number of clusters K.

  We can observe from "WEDDING 0" video with an expected $K = 4$ clusters, that the eigengap information is not as relevant as for easy dataset.

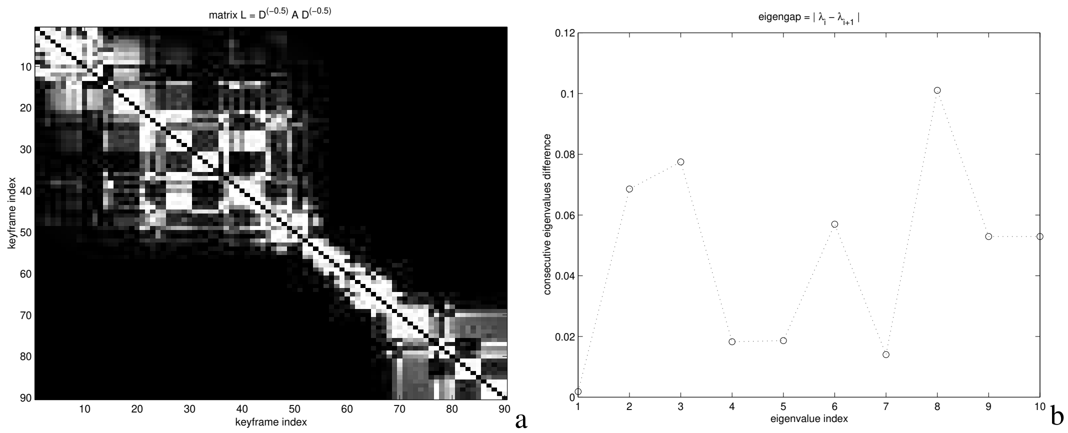


Figure 6.10: Clustering "WEDDING 0" video (a) Affinity matrix (b) Eigengap

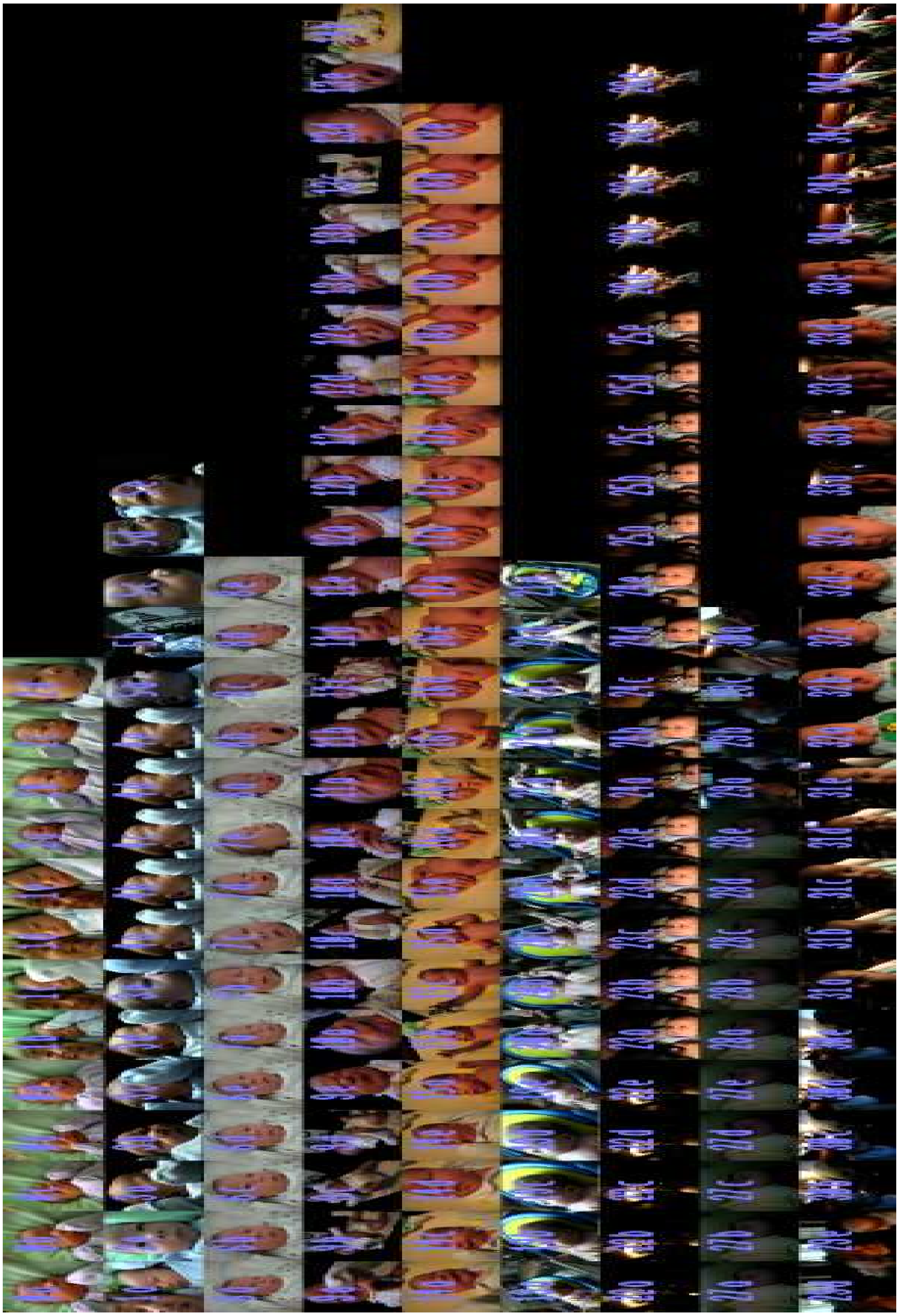


Figure 6.11: global visualization of RYLAN 0 video

Name	Video-clip	Duration	Shots	Clusts	COfs	SIEfs	COvs	SIEvs	SIEvsk
$CINDY_0$	V_0	20:01	15	4	1	1	1	1	0
$CINDY_1$	V_1	21:17	19	2	0	0	0	0	0
$CINDY_2$	V_2	18:21	18	8	4	5	5	5	2
$CINDY_3$	V_3	20:12	19	5	3	3	3	3	2
$CINDY_4$	V_4	21:39	18	6	3	4	3	4	3
$CINDY_5$	V_5	19:47	18	5	4	5	4	4	2
$CLEM_0$	V_6	20:01	47	14	14	16	10	13	13
$CLEM_1$	V_7	20:01	59	15	16	25	17	23	19
$CLEM_2$	V_8	20:01	54	10	13	28	11	19	15
$CLEM_3$	V_9	22:45	62	7	7	18	7	15	9
$CLEM_4$	V_{10}	20:01	35	6	5	7	4	5	5
$CLEM_5$	V_{11}	18:52	48	7	6	15	6	14	14
SUE_0	V_{12}	23:01	8	3	1	1	1	1	1
SUE_1	V_{13}	25:01	11	4	0	0	0	0	0
SUE_2	V_{14}	20:01	7	4	1	1	1	1	1
SUE_4	V_{15}	20:02	10	5	1	1	1	1	1
SUE_5	V_{16}	20:01	8	3	2	4	2	4	2
$THERESA_0$	V_{17}	20:01	19	5	5	7	2	3	3
$THERESA_1$	V_{18}	18:02	4	1	0	0	0	0	0
$WEDDING_0$	V_{19}	23:57	18	4	4	10	3	6	6
$CHARLIE_0$	V_{20}	23:07	105	19	14	22	16	26	21
$ELENI_4$	V_{21}	20:49	10	4	2	3	2	2	2
$BUBBLES_0$	V_{22}	19:56	12	4	0	0	0	0	0
$RYLAN_0$	V_{23}	20:27	35	9	2	6	1	1	1
$CARO_1$	V_{24}	20:00	18	5	2	3	2	2	2
$SUZANNE_2$	V_{25}	20:00	12	7	4	5	2	2	2
EVA_2	V_{26}	20:00	54	6	4	9	4	9	9
$CARO_4$	V_{27}	20:00	16	6	1	1	1	1	1
EVA_4	V_{28}	20:00	20	5	3	6	3	4	4
$CARO_3$	V_{29}	20:00	22	6	5	8	3	4	3
Total		617:34	801	192	202				

Table 6.4: Video Clustering Results on Kodak Consumer Video Database.

Chapter 7

Concluding remarks on the project

Throughout this study, the whole video structuring process was examined. The first step, shot boundary detection, was object-oriented designed using both motion estimation-based analysis and color histogram-based analysis. Before clustering video shots, we saw that the video shot feature extraction process is fundamental. Section 5 proposes a solution combining color distributions with a time-constraint to the two problems of perceptual grouping algorithms: (i) how to compute pairwise similarity elements and (ii) how to aggregate these elements. The key part of my mission as a real contribution in video analysis research area was the implementation of a video shot clustering algorithm using a spectral method. The results are encouraging and these methods will be exploited as a pre-processor for object/event detection. The goal of the intern-ship as a development of tools for video segmentation at the state-of-the-art of video analysis current research was reached. All the programs were implemented in C, C++ and Matlab language under Linux and Unix. They are available in the common directory[1] of the vision group with READMEs, two of them described in appendix. The software was developed in a way that adding new features or varying parameters is easily achieved.

As inside a research environment, I assisted to seminars not only in computer vision area but also in statistical machine learning and speech recognition. During ICME conference[2] in Lausanne, I met Dr. Hanjalic, referenced twice in this report ([**?**, **?**]), with who I could exchange ideas in the area of multimedia indexing. At the end of the day, a special session was organized with, in particularly, John Smith and Malcom Slaney from IBM. This last researcher published recently on discussing multimedia limitation in terms of technical research [**?**]. As future work in this domain, the converging research idea is to mix modalities: speech, text and image/video: That is what IDIAP is currently investigating[3]. One of the ideas for future work that can be generated from this project is the fusion of the two shot boundary detectors. Then K and σ parameters from the video shot clustering model selection are not automatically guessed: this is still an open area. Also all the programs woulod need to be push up on a same platform, like Torch[4] for instance, with a nice user interface to facilitate ineraction.

Finally, at Idiap, researchers and PhD students come from all over the world (from Australia to Mexico, China, Canada, Russia, India, Malaysia, Greece etc.): I appreciated this mix of cultural backgrounds.

[1] in /home/vision/common/Video-Structuring/

[2] http://www.icme2002.org

[3] IM^2, http://www.im2.ch/

[4] http://www.torch.ch

Chapter 8

Appendix

Why is the main algorithm from Chapter 6 is called spectral ?

Let A be an $n \times n$ matrix and consider the vector equation

$$A\vec{v} = \lambda\vec{v}, \tag{8.1}$$

where λ is a scalar value.

It is clear that if $\vec{v} = \vec{0}$, we have a solution for any value of λ. A value of for which the equation has a solution with $\vec{v} \neq \vec{0}$ is called an *eigenvalue* or *characteristic value* of the matrix . The corresponding solutions are called *eigenvectors* or *characteristic vectors* of A. In the problem above, we are looking for vectors that when multiplied by the matrix A, give a scalar multiple of itself.

The set of eigenvalues of A is commonly called the *spectrum* of and the largest of the absolute values of the eigenvalues is called the *spectral radius* of A.

Bibliography