

DELIVERABLE SUBMISSION SHEET

To: Claude Poliart *(Project Officer)*
 EUROPEAN COMMISSION
 DG Info E3, Cultural heritage and technology enhanced learning
 EUFO 1154
 Rue Alcide de Gasperi
 L-2920 Luxembourg

From: Project name: Enabling Access to Sound Archives through Integration,
 Enrichment and Retrieval

Project acronym: EASAIER Project number: 033902

Person: Joshua Reiss

Organisation: Queen Mary University of London

Date: 15 August 2008

The following deliverable:

Deliverable title: Prototype on cross media retrieval system

Deliverable number: D.3.3

is now complete. It is available for your inspection.
 Relevant descriptive documents are attached.
 1 copy herewith.
 2 copies herewith

} *Tick all that
apply*

The deliverable is: on paper
 × on
 (url:www.easaier.org)

For all paper deliverables, and other deliverables as appropriate:

Date of del: 15 August 2008 Version: 1
 Author: Ivan Damnjanovic No. of pages: 41
 Status: Public Restricted Confidential *(tick one)*

Sent electronically to Claude Poliart	Functional mail box	Claude.Poliart@ec.eur opa.eu	ON DATE	15/08/08
ATTESTATION				
I confirm that this electronic version is the same as the paper version submitted herewith.				
NAME	Betty Woessner	ORGANISATION	QMUL	
SIGNATURE				
DATE	15 August 2008			



D3.3

Prototype on cross-media retrieval system

Abstract

Deliverable 3.3, Prototype on cross-media retrieval system, describes components developed in EASAIER project that implement retrieval across media types, including related text, images and video materials. The document details cross-media retrieval systems and their prototypes, describes their functionality, the challenges and the underlying technologies, evaluation of the prototypes, and the level of integration of the retrieval systems that has so far been achieved. The focus was given on integration of the components that presents to the user web and archive materials related to his or her query, as well as video analysis components that will further enhance user experience through faster inter- and intra-related video browsing. This deliverable also presents the initial user evaluation results on the EASAIER cross-media retrieval engine.

Version: Final Draft

Date: 12 August 2008

Editor: QMUL

Contributors: QMUL, STI, AKKA, RSAMD

Table of Contents

1.	DOCUMENT HISTORY	3
2.	EXECUTIVE SUMMARY	4
2.1.	PROJECT SUMMARY	4
2.2.	DELIVERABLE SUMMARY	4
3.	USER NEEDS AND REQUIREMENTS	6
3.1.	INTRODUCTION.....	6
3.2.	EVALUATION METHODOLOGY AND METRICS	6
4.	STATE OF THE ART	8
4.1.	WEB RELATED MEDIA	8
4.2.	VIDEO ANALYSIS.....	10
4.2.1.	<i>Pixel Comparison.....</i>	<i>10</i>
4.2.2.	<i>Block-based comparison</i>	<i>11</i>
4.2.3.	<i>Histogram comparison.....</i>	<i>12</i>
4.2.4.	<i>Global Histogram Comparison.....</i>	<i>13</i>
4.2.5.	<i>Local Histogram Comparison.....</i>	<i>15</i>
4.2.6.	<i>Clustering-Based Temporal Video Segmentation</i>	<i>16</i>
4.2.7.	<i>Feature Based Temporal Video Segmentation.....</i>	<i>16</i>
4.2.8.	<i>Model Driven Temporal Video Segmentation</i>	<i>16</i>
5.	CROSS-MEDIA RETRIEVAL INTEGRATION.....	19
5.1.	THE EASAIER SYSTEM ARCHITECTURE	19
5.2.	WEB-RELATED MEDIA WORKFLOW	19
5.3.	WEB-RELATED CONTENT RETRIEVAL INTEGRATION.....	22
6.	WEB RELATED CONTENT RETRIEVAL COMPONENTS	24
7.	RELATED MEDIA IN THE ARCHIVE.....	26
7.1.	RELATED MEDIA WORKFLOW	26
7.2.	INTEGRATION IN THE SOUND ACCESS CLIENT APPLICATION	27
8.	VIDEO ANALYSIS USING SPECTRAL CLUSTERING TO SUPPORT CROSS-MEDIA QUERIES	28
8.1.	SPECTRAL CLUSTERING FOR SHOT BOUNDARY DETECTION	29
8.2.	SHOT BOUNDARY DETECTION AND KEY FRAME EXTRACTION USING RECURSIVE NORMALIZED CUTS.....	30
8.3.	EXPERIMENTAL EVALUATION.....	34
9.	USER EVALUATION.....	37
9.1.	DISCUSSION	38
9.2.	SUMMARY AND CONCLUSION	39
10.	REFERENCES	40

2. Executive Summary

2.1. *Project summary*

The two and a half year European project, Enabling Access to Sound Archives through Integration, Enrichment and Retrieval (EASAIER) allows archived materials to be accessed in different ways and at different levels. The system is designed with libraries, museums, broadcast archives, and music schools and archives in mind. However, the tools may be used by anyone interested in accessing archived material; amateur or professional, regardless of the material involved. Furthermore, it enriches the access experience as well, since it enables the user to experiment with the materials in exciting new ways. EASAIER implements recent advances in machine learning, music and speech processing, and information retrieval. Furthermore, it addresses a growing demand for interactive electronic materials. By enriching and organizing the materials in the archive, we will create new and innovative methods of access. This enhanced access stimulates use, because multimedia resources will be connected, and archived material will be exploited and made visible to the right users. This will be deployed in connection with strong evaluation studies using communities of users and content managers.

2.2. *Deliverable summary*

This deliverable presents work that has been done on cross-media retrieval in the EASAIER project. The work has been two-fold. First, we focused on exploitation of the research performed in other work packages, namely WP2 Media Semantics and Ontologies and WP6 Intelligent Interfaces, to present the user images, videos and text materials related to his search, no matter if they are stored in archive or collected from the World Wide Web. Second, we developed video analysis components to support video related cross-media queries. All audio analysis done on sound files is also done on video files. In addition, shot detection and key-frame extraction components will support visual similarity that can be combined with audio similarity retrieval. Extracted key-frames can be also presented to the user when enabling him to browse faster through the video.

In the following section, a short overview of user requirements studies is given. This is the work that has been mainly done in beginning of the project and presented in part in deliverable 3.1, but nevertheless refined during the project. It also presents evaluation methodology and metrics defined in WP7 and used in subjective evaluation of the cross-media retrieval system. In section 4, the state-of-the-art in the field of web related media retrieval is presented, followed by a thorough study of shot boundary detection and key frame extraction that led to development of particular components.

Integration of cross-media retrieval components in EASAIER architecture is explained in section 5. The cross-media functionality is integrated in the components “Related media class” and “Web related media class” in the main Access service. The “Related media class” uses the Sparql endpoint and rdf storage to retrieve related media in the current archive. The “Web related media class” uses web-services providing by the World Wide Web. Sections 6 and 7 give further description of the Web and archive related materials components, respectively.

A novel spectral clustering approach to video analysis is presented in section 8. Combined with audio features, this component will enhance search and retrieval of related videos and also enables the user to browse faster and find particular video section of her/his interests. We also present experimental evaluation of the component at the end of the section.

Finally, section 9 discusses feedback from the student evaluators provided in their written evaluations of the client from April 2008, focussing on their comments relating to performance,

public

usefulness and usability of the cross-media retrieval system. Testing was performed on an Alpha release of the client and, as such, evaluation of some procedures was not possible. However, the users comments are enlightening and offer some useful guidance about their contexts and uses for future development of cross-media components in EASAIER.

3. User needs and requirements

3.1. Introduction

Initial user needs and requirements were derived from the literature, as discussed in D3.1. One of the outcomes of this review indicated the need for web-based access, integration of other media, and enriched access and playback tools.

The HOTBED project [1] identified a strong user need for other media in addition to audio. The JISC User Requirements Study for a Moving Pictures and Sound Portal [2] also identified the need to have “cross-searching between still and time-based collections.” Furthermore, the HOTBED study identified the use of video as having a strong impact in aural learning. Thus, it is clear that this collection must incorporate video and other media as well as audio, and provide significant interaction between the media.

As discussed in D3.1, collating the findings from these studies, we have established the potential value of a cross-media information retrieval system as part of the larger EASAIER system. This would allow the user to access a range of different types of media derived from a search query. A possible use case scenario follows:

Use-Case Scenario - The amateur musician

Greg is guitarist in a band consisting of old school friends. A vinyl enthusiast, the pride of Greg’s collection is a complete set of Pink Floyd and Led Zeppelin albums. On borrowing his girlfriend’s MP3 player, Greg discovered the large amount of material available digitally. Hearing an alternative version of Led Zep’s Black Dog on a late night radio show, Greg became interested in finding alternative and live recordings of the songs played by his rock’n’roll heroes, but he finds it difficult searching the internet for such tracks as he can rarely listen to them without buying them first.

Following a Google search, Greg logs onto a classic rock’n’roll archive that uses the EASAIER system. He enters ‘Pink Floyd’ in the author/title field and also puts ‘live’ in the keyword field. To his delight, his search returns an alternative, live version of a segment from Atom Heart Mother with a pared-down orchestration – much sparser than the original studio mix. The metadata displays the distinctive cover art of the Atom Heart Mother album (showing a large cow named Lulubelle III) and a picture of a Pink Floyd performance from around the time the album came out. Happy with his musical research, Greg plays this segment again, and looks for further alternative versions of his favourite tracks. Greg sends his fellow band members an email with a link to the website employing the EASAIER system.

3.2. Evaluation methodology and metrics

As was discussed in D7.1, it is central to the development of the EASAIER system that user needs are satisfied by means of a process of formative and summative evaluations. These evaluations should focus on Performance, Usability and Usefulness [3]. Two key user groups have been identified, the music user and the content manager. The music users, represented by a group of ten classical music students at RSAMD, have been closely involved in evaluations of the client, through direct feedback, focus groups, observations and their own written evaluations following these agreed criteria [3]:

- Performance: Precision, Recall, Relevance, Response Time
- Usability: Ease of use, Aesthetics, Navigation, Terminology, Learnability
- Usefulness: Relevance, Format, Reliability, Level, Coverage

These evaluations are discussed in detail in [4], [5] and key elements are extracted from the original evaluation texts for this report.

The other user group, the EUAB, have been consulted in depth about their needs and requirements and regular contact has kept this communication open and up-to-date.

Challenge	Agreed criteria and mechanisms
Objectives	Investigate performance, usability and usefulness of cross-media information retrieval system for expert musicians, but non-expert users and identify improvements.
User groups and target audiences	Ten classical music students at RSAMD.
Testing and Evaluation methods to be used	User group written evaluations.
Reporting/discussion mechanisms	10 students submitted evaluations of around 2,500 words. The data were analysed in the context of a framework which focussed on aspects of performance, usability and usefulness. Key comments on cross-media retrieval were extracted.
Schedule	Students had access to the client on a dedicated laptop for a period of four weeks. Their work was submitted on Mon Apr 28 2008.

4. State of the Art

4.1. Web related media

A web mashup is a website or a web application that aggregates data from third party content or service providers on the web, creating a new integrated tool or service. This involves two or more sources. Mashups facilitate information augmentation and integration through simple methods for the developer. The combination of data with other useful and related data reduces navigation and increases information gain for the user [6]. Typically a request is posed by the mashup service to an external content provider, relying on a certain predefined message exchange protocol, such as REST, SOAP, or RSS. Next, the content provider transmits the desired result in a certain format, such as XML, JSON or plain HTML.

On the side of the mashup endpoint, all inputs are then standardized, combined, and then the final output is displayed in the user interface. This means that all the information from 3rd party sources is connected on an application level rather than on a data level.

The emergence of large amounts of user-generated content on the web is resulting in more data and metadata becoming available, and a lot of content providers publish their data in a way that is accessible for aggregation by external parties. Programmableweb.com lists a number 868 available APIs for public access. However, the majority of websites don't expose their content through an official API (one example is Wikipedia). In such cases it is possible to apply screen scraping, i.e. the retrieval of the content of one or multiple webpages (usually the raw HTML sources) over HTTP and parsing it in order to extract the desired information. In addition, local resources, for example contents from a database, can also be part of a mashup. This further increases the amount of possible data sources for mashups.

Architecture

The three roles in a mashup are the data providers, the mashup site, and the consumer.

Data provider

In a mashup scenario, the data provider offers its contents through an API over a certain protocol such as HTTP or SOAP, but the data can also come from the screen scraping of a website or from local data sources. If an API is available, the data provider often provides documentation and usage examples for the developer community.

The incentives of websites to publish their data can be the branding of the product name and gaining visibility, the placement of advertisements, or that the service is made available on a pay-per-use basis (e.g. www.salesforce.com) [7]. Often, the mashup developer needs to register a free product key in order to access the API. The developer can be kept up to date over developments with the API.

Mashup site

At this level, all the data received by the content providers is combined into a single service or application. The *client-based model* uses client-side scripting languages such as Javascript or AJAX to assemble and integrate the content from the original source directly inside the client's web browser. With the *server-based model*, the data from the content provider is further processed by an intermediate web server component, using a scripting language such as JSP, ASP, etc. to assemble the final output. There are also mashup applications are a combination of both approaches.

Some data providers modify their APIs and the existing solution implemented by the mashup site may not be supported anymore. While deprecated version of the most popular content

public

providers' APIs usually remain supported for legacy reasons, in practice this often creates a problem if the content stems from screenscraping and when the structure of such a source is modified. The mashup site should implement safeguards to handle situations that a source becomes unavailable, rather than having the whole mashup application not working.

The mashup application itself may also provide a public API and become a content provider.

Consumer

The consumer typically accesses the mashup using a web browser. The web browser executes client-side logic, often using JavaScript or AJAX. AJAX allows for asynchronous communication between the browser and the server, typically relying on XML. This enables interactivity with the web application without refreshing the browser. An AJAX call only changes parts of the HTML web pages.

Mashup Content Types and Presentation

There exist different types of mashups, depending on the data sources. The majority of mashups are (or include) mapping mashups, see Figure 4.1. Those applications typically combine a visual map that can be browsed by the browser (e.g. Google Maps) with data that contains location based data. One example is a Housingmaps.com, a mashup of Craigslist and Google maps for browsing rental listings on the map. This is a good example of how to bring two large-scale applications together, creating a useful application in a relatively simple manner.

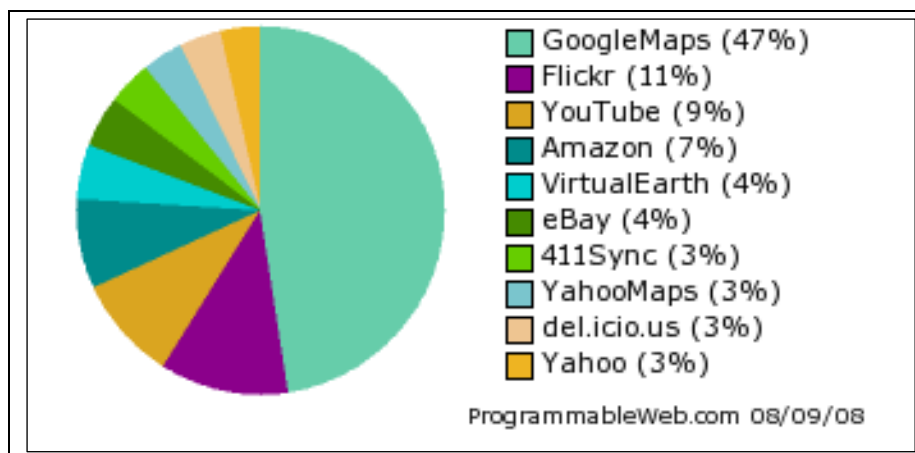


Figure 4.1 Top mashup APIs.

Other types include the aggregation of multimedia sources (e.g. Flickr, YouTube) or news feeds from multiple sources. A lot of news sources syndicate their content via RSS/Atom, and the user aggregates those news entries from a personalized selection of sources. In a similar fashion, it is possible to aggregate shopping data from commercial providers such as Amazon, eBay, or information from the travelling domain.

The type of the data sources that are integrated by a mashup application plays a big role in the combination of the different sources and how to visualize the final output. There exist at least two ways how to combine and display diverse data sources in a mashup visualization [8]: either by using multiple displays assembled in a grid-like way or a single display that combines all the sources. With a single display of all the content, it's easier to display relationships between items, however it depends on the type of content and structure of the sources to what extent they can be related. For mapping mashups the combination of the sources is typically done by relating the location based data of the one source with the mapping service, matching for instance by the

public

address field or the zipcode. The grid layout assembles content from multiple sources into a single page – the user doesn't need to formulate queries at different web portals or browse websites, however the relationship between the content is limited.

In the scope of the EASAIER project, we apply such a mashup application in order to accompany and enrich the audio archives' contents stored in the EASAIER system with additional related media from the web. The Web Related Media component also provides the user with a direct access point to the web to further investigate related content about the resources from the archive (e.g. related artists). We currently focus on multimedia sources, such as pictures and videos, and we also include information about commercial offers from online shops that sell items related to the audio asset.

4.2. Video analysis

As the first step of video analysis we focus on temporal video segmentation. The temporal video segmentation research efforts have resulted in a great variety of algorithms. Early work focused on cut detection, while more recent techniques dealt with the more difficult problem of gradual-transition detection. Fades, dissolves and wipes are special video editing effects that gradually change the content and therefore are more difficult to detect. Fade-in is an editing effect which allows the progressive transition from a solid black frame to full brightness of a shot content, while a fade-out is a progressive darkening of a shot until the last frame becomes completely black. Dissolve is a superimposition of a fade-in and a fade-out: the first shot fades-out while the following fades-in to full brightness. Wipe is a content transition from one scene to another wherein the new scene is revealed by a moving line or pattern. In its simplest form, it simulates a window shade being drawn. More sophisticated variations include colorized wipes, quivering wipes and triangle wipes. In the following sections a number of relevant methods is described and compared with the algorithm proposed. The majority of algorithms for temporal video segmentation exploit uncompressed video data. Usually, a similarity measure between successive images is defined. When two images are sufficiently dissimilar, there is a high probability of a cut. Gradual transitions are detected by using cumulative difference measures and more sophisticated thresholding schemes. Based on the metrics that is used to detect the difference between successive frames, the algorithms for temporal video segmentation in uncompressed domain can be divided broadly into three categories: pixel, block-based and histogram comparisons.

4.2.1. Pixel Comparison

Pair-wise pixel comparison (also called template matching) evaluates the differences in intensity or colour values of corresponding pixels in two successive frames. The simplest way is to calculate the absolute sum of pixel differences and compare it against a threshold [10]:

$$D(i, i+1) = \frac{\sum_{x=1}^X \sum_{y=1}^Y |P_i(x, y) - P_{i+1}(x, y)|}{X \cdot Y} \quad (4.1)$$

for grey level images, and

$$D(i, i+1) = \frac{\sum_{x=1}^X \sum_{y=1}^Y \sum_c |P_i(x, y, c) - P_{i+1}(x, y, c)|}{X \cdot Y} \quad (4.2)$$

for colour images, where i and $i+1$ are two successive frames with dimension $X \cdot Y$, $P_i(x, y)$, is the intensity value of the pixel at the coordinates (x, y) , in frame i , c is index for the colour components and $P_i(x, y, c)$, is the colour component of the pixel at y, x in frame i .

public

A cut is detected if the difference $D(i,i+1)$ is above a pre specified threshold T . The main disadvantage of this method is that it is not able to distinguish between a large change in a small area and a small change in a large area. For example, cuts are misdected when a small part of the frame undergoes a large, rapid change. Therefore, methods based on simple pixel comparison are sensitive to object and camera movements. A possible improvement is to count the number of pixels that change in value more than some threshold and to compare the total against a second threshold [11] [12]:

$$DP(i,i+1,x,y) = \begin{cases} 1 & \text{if } |P_i(x,y) - P_{i+1}(x,y)| > T_1, \\ 0, & \text{otherwise,} \end{cases} \quad (4.3)$$

$$D(i,i+1) = \frac{\sum_{x=1}^X \sum_{y=1}^Y DP(i,i+1,x,y)}{X \cdot Y} \quad (4.4)$$

If the percentage of changed pixels $D(i,i+1)$ is greater than a threshold T_2 , a cut is detected. Although some irrelevant frame differences are filtered out, these approaches are still sensitive to object and camera movements. For example, if camera pans, a large number of pixels can be judged as changed, even though there is actually a shift with a few pixels. It is possible to reduce this effect to certain extend by the application of a smoothing filter: before the comparison each pixel is replaced by the mean value of its neighbours.

4.2.2. Block-based comparison

In contrast to template matching that is based on global image characteristic (pixel by pixel differences), block-based approaches use local characteristic to increase the robustness to camera and object movement. Each frame i is divided into b blocks that are compared with their corresponding blocks in $i+1$. Typically, the difference between i and $i+1$ is measured by

$$D(i,i+1) = \sum_{k=1}^b c_k \cdot DP(i,i+1,k) \quad (4.5)$$

where c_k is a predetermined coefficient for the block k and $DP(i,i+1,k)$ is a partial match value between the k th blocks in i and $i+1$ frames.

In [13] corresponding blocks are compared using a likelihood ratio:

$$\lambda_k = \frac{\left[\frac{\sigma_{k,i} + \sigma_{k,i+1}}{2} + \left(\frac{\mu_{k,i} + \mu_{k,i+1}}{2} \right)^2 \right]^2}{\sigma_{k,i} \cdot \sigma_{k,i+1}} \quad (4.6)$$

where $\sigma_{k,i}, \sigma_{k,i+1}$ are the mean intensity values for the two corresponding blocks k in the consecutive frames i and $i+1$, and $\mu_{k,i}, \mu_{k,i+1}$ are their variances, respectively. Then, the number of blocks for which the likelihood ratio is greater than a threshold T_1 is counted:

$$DP(i,i+1,x,y) = \begin{cases} 1 & \text{if } \lambda_k > T_1, \\ 0, & \text{otherwise,} \end{cases} \quad (4.7)$$

A cut is declared when the number of changed blocks is large enough, i.e. $D(i,i+1)$ is greater than a given threshold T_2 and $c_k=1$ for all k . Compared to template matching, this method is more tolerant to slow and small object motion from frame to frame. On the other hand, it is slower due to the complexity of the statistical formulas. Additional potential disadvantage is that no change will

public

be detected in the case of two corresponding blocks that are different but have the same density function. Such situations, however, are very unlikely.

Another block-based technique is proposed by Shahraray [14]. The frame is divided into 12 non-overlapping blocks. For each of them the best match is found in the respective neighbourhoods in the previous image based on image intensity values. A non-linear order statistics filter is used to combine the match values. Thus, the effect of camera and object movements is further suppressed. The author claims that such similarity measure of two images is more consistent with human judgement. Both cuts and gradual transitions are detected. Cuts are found using thresholds like in the other approaches that are discussed while gradual transitions are detected by identifying sustained low-level increase in match values.

Xiong, Lee and Ip [15] describe a method they call *net comparison*, which attempts to detect cuts inspecting only part of the image. It is shown that the error will be low enough if less than half of so called base windows (non-overlapping square blocks, as in Figure 4.2) are checked. Under an assumption about the largest movement between two images, the size of the windows can be chosen large enough to be indifferent to a non-break change and small enough to contain the spatial information as much as possible. Base windows are compared using the difference between the mean values of their grey-level or colour values. If this difference is larger than a threshold, the region is considered changed. When the number of changed windows is greater than another threshold, a cut is declared.

The experiments demonstrated that the approach is faster and more accurate than pixel pairwise, likelihood and local histogram methods. In their subsequent paper [16], the idea of video subsampling into space is further extended to subsampling in both space and time. The new Step variable algorithm detects both abrupt and gradual transition comparing frames i and j , where $j=i+myStep$. If no significant change is found between them, the move is with half step forward and the next comparison is between $i+myStep/2$ and $j+myStep/2$. Otherwise, binary search is used to locate the change. If i and j are successive and their difference is bigger than a threshold, cut is declared.

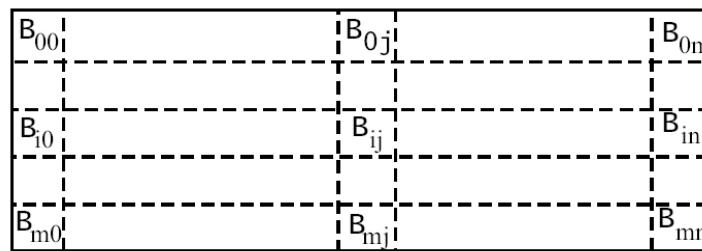


Figure 4.2 Non-overlapping square blocks in net comparison algorithm

Otherwise, edge differences between the two frames are compared against another threshold to check for gradual transition. Obviously, the performance depends on the proper setting of $myStep$: large steps are efficient but increase the number of false alarms, too small steps may result in missing gradual transition. In addition, the approach is very sensitive to object and camera motion.

4.2.3. Histogram comparison

A further step towards reducing sensitivity to camera and object movements can be done by comparing the histograms of successive images. The idea behind histogram-based approaches is that two frames with unchanging background and unchanging (although moving) objects will have little difference in their histograms. In addition, histograms are invariant to image rotation and change slowly under the variations of viewing angle and scale. As a disadvantage one can note that

public

two images with similar histograms may have completely different content. However, the probability for such events is sufficiently low. Moreover, techniques for dealing with this problem have already been proposed in [17].

A grey level (colour) histogram of a frame i is an n -dimensional vector $H_i(j)=1, \dots, n$ where n is the number of grey levels (colours) and $H(j)$ is the number of pixels from the frame i with grey level (colour) j .

4.2.4. Global Histogram Comparison

The simplest approach uses an adaptation of the metrics from Equation (4.1): instead of intensity values, grey level histograms are compared. A cut is declared if the absolute sum of histogram differences between two successive frames $D(i, i+1)$ is greater than a threshold T :

$$D(i, i+1) = \sum_{j=1}^n |H_i(j) - H_{i+1}(j)| \quad (4.8)$$

where $H_i(j)$ is the histogram value for the grey level j in the frame i , j is the grey value and n is the total number of grey levels.

Another simple and very effective approach is to compare colour histograms. Zhang, Kankanhalli and Smoliar [11] apply Equation (4.8) where j , instead of grey levels, denotes a code value derived from the three colour intensities of a pixel. In order to reduce the bin number (3 colours \times 8 bits create histograms with 2^{24} bins), only the upper two bits of each colour intensity component are used to compose the colour code. The comparison of the resulting 64 bins has been shown to give sufficient accuracy.

To enhance the difference between two frames across a cut, several authors propose the use of the χ^2 test to compare the (colour) $H_i(j)$ histograms and $H_{i+1}(j)$ of the two successive frames i and $i+1$:

$$D(i, i+1) = \sum_{j=1}^n \frac{|H_i(j) - H_{i+1}(j)|^2}{H_{i+1}(j)} \quad (4.9)$$

When the difference is larger than a given threshold T , a cut is declared. However, experimental results reported in [11] show that χ^2 test not only enhances the difference between two frames across a cut but also increases the difference due to camera and object movements. Hence, the overall performance is not necessarily better than the linear histogram comparison represented in Equation (4.9). In addition, χ^2 statistics requires more computational time. Gargi *et al.* [18] evaluate the performance of three histogram based methods using six different colour coordinate systems: RGB, HSV, YIQ, $L^*a^*b^*$, $L^*u^*v^*$ and Munsell. The RGB histogram of a frame is computed as three sets of 256 bins. The other five histograms are represented as a 2-dimensional distribution over the two non-intensity based dimensions of the colour spaces, namely: H and S for the HSV, I and Q for the YIQ, a^* and b^* for the $L^*a^*b^*$, u^* and v^* for the $L^*u^*v^*$ and hue and chroma components for the Munsell space. The number of bins is 1600 (40 \times 40) for the $L^*a^*b^*$, $L^*u^*v^*$ and YIQ histograms and 1800 (60 hues \times 30 saturations/chroma) for the HSV and Munsell space histograms. The difference functions used to compare histograms of two consecutive frames are defined as follows:

$$\text{Bin-to-bin differences:} \quad D(i, i+1) = \sum_{j=1}^n |H_i(j) - H_{i+1}(j)| \quad (4.10)$$

Histogram intersection:
$$D(i, i+1) = 1 - \frac{\sum_{j=1}^n \min(H_i(j) - H_{i+1}(j))}{\sum_{j=1}^n \max(H_i(j) - H_{i+1}(j))} \quad (4.11)$$

Note that for two identical histograms the intersection is 1 and the difference 0 while for two frames which do not share even a single pixel of the same colour (bin), the difference is 1.

Weighted bin differences:
$$D(i, i+1) = \sum_{j=1}^n \sum_{k \in N(k)} W(k) \cdot (H_i(j) - H_{i+1}(j)) \quad (4.12)$$

where $N(k)$ is a neighbourhood of bin j and $W(k)$ is the weight value assigned to that neighbour. A 3x3 or 3 neighbourhoods are used in the case of 2-dimensional and 1-dimensional histograms, respectively.

It is found that in terms of overall classification accuracy YIQ, L*a*b* and Munsell colour coordinate spaces perform well, followed by HSV, L*u*v* and RGB. In terms of computational cost of conversion from RGB, the HSV and YIQ are the least expensive, followed by L*a*b*, L*u*v* and the Munsell space.

So far only histogram comparison techniques for cut detection have been presented. They are based on the fact that there is a big difference between the frames across a cut that results in a high peak in the histogram comparison and can be easily detected using one threshold. However, such single threshold approaches are not suitable to detect gradual transitions. Although during a gradual transition the frame-to-frame differences are usually higher than those within a shot, they are much smaller than the differences in the case of cut and cannot be detected with the same threshold. On the other hand, object and camera motions might entail bigger differences than the gradual transition. Hence, lowering the threshold will increase the number of false positives. Below we review a simple and effective two-thresholds technique for gradual transition recognition.

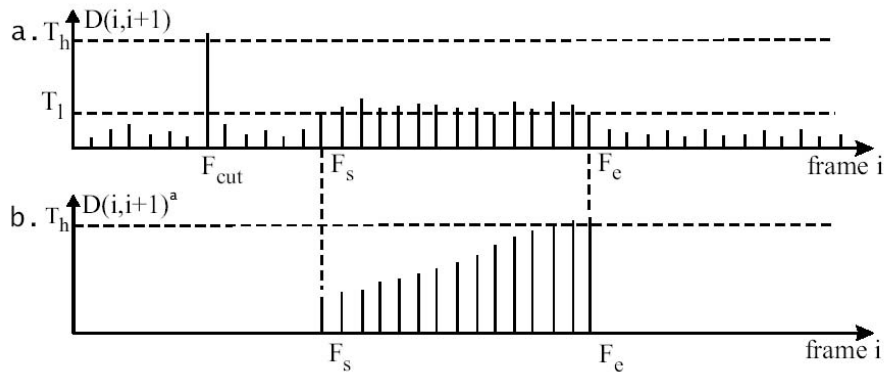


Figure 4.3 Twin comparison: a. consecutive, b. accumulated histogram differences. Figure taken from [11].

The *twin-comparison* method [19] takes into account the cumulative differences between frames of the gradual transition. In the first pass a high threshold T_h is used to detect cuts as shown in Figure 4.3 (a). In the second pass a lower threshold T_l is employed to detect the potential starting frame F_s of a gradual transition. F_s is then compared to subsequent frames (Figure 4.3 (b)). This is called an accumulated comparison as during a gradual transition this difference value increases. The end frame F_e of the transition is detected when the difference between consecutive frames decreases to less than T_l , while the accumulated comparison has increased to a value higher than T_h . If the consecutive difference falls below T_l before the accumulated difference exceeds T_h , then the potential start frame F_s is dropped and the search continues for other gradual transitions. It was found, however, that there are some gradual transitions during which the consecutive difference

public

falls below the lower threshold. This problem can be easily solved by setting a tolerance value that allows a certain number of consecutive frames with low difference values before rejecting the transition candidate. As it can be seen, the twin-comparison detects both abrupt and gradual transitions at the same time. Boreczky and Rowe [20] compared several temporal video segmentation techniques on real video sequences and found that twin-comparison is a simple algorithm that works very well.

4.2.5. Local Histogram Comparison

As it was already discussed, histogram based approaches are simple and more robust to object and camera movements but they ignore the spatial information and, therefore, fail when two different images have similar histograms. On the other hand, block based comparison methods make use of spatial information. They typically perform better than pair-wise pixel comparison but are still sensitive to camera and object motion and are also computationally expensive. By integrating the two paradigms, false alarms due to camera and object movement can be reduced while enough spatial information is retained to produce more accurate results.

The frame-to-frame difference of frame i and frame $i+1$ is computed as:

$$D(i, i+1) = \sum_{k=1}^b DP(i, i+1, k) \quad (4.13)$$

$$DP(i, i+1, k) = \sum_j^n |H_i(j, k) - H_{i+1}(j, k)| \quad (4.14)$$

where $H_i(j, k)$ denotes the histogram value at grey level j for the region (block) k and b is the total number of the blocks.

For example, Nagasaka and Tanaka [12] compare several statistics based on grey-level and colour pixel differences and histogram comparisons. The best results were obtained by breaking the image into 16 equal-sized regions, using χ^2 test on colour histograms for these regions and discarding the largest differences to reduce the effects of noise, object and camera movements.

Another approach based on local histogram comparison is proposed by Swanberg *et al.* [21]. The partial difference $DP(i, i+1, k)$ is measured by comparing the colour RGB histograms of the blocks using the following equation:

$$DP(i, i+1, k) = \sum_{c \in \{R, G, B\}} \sum_{l=1}^n \frac{(H_i^c(l) - H_{i+1}^c(l))^2}{H_i^c(l) - H_{i+1}^c(l)} \quad (4.15)$$

Then, Equation (4.5) is applied where c_k is $1/b$ for all k .

Lee and Ip [22] introduce a *selective HSV histogram* comparison algorithm. In order to reduce the frame-to-frame differences caused by change in intensity or shade, image blocks are compared in HSV (hue, saturation, value) colour space. It is the use of hue that makes the algorithm insensitive to such changes since hue is independent of saturation and intensity. However, as hue is unstable when the saturation or the value is very low, selective comparison is proposed. To further improve the algorithm by increasing the differences across a cut, local histogram comparison is performed. It is shown that the algorithm outperforms both histogram (grey level global and local) and pixel differences based approaches. However, none of the algorithms gives satisfactory performance on very dark video images.

4.2.6. Clustering-Based Temporal Video Segmentation

The approaches discussed so far rely on suitable thresholding of similarities between successive frames. However, the thresholds are typically highly sensitive to the type of input video. This drawback is overcome by the application of *unsupervised clustering* algorithm. More specifically, the temporal video segmentation is viewed as a 2-class clustering problem ("scene change" and "no scene change") and the well-known K-means algorithm [23] is used to cluster frame dissimilarities. Then the frames from the cluster "scene change" which are temporary adjacent are labelled as belonging to a gradual transition and the other frames from this cluster are considered as cuts. Two similarity measures based on colour histograms were used: χ^2 statistics and the histogram difference defined in Equation (4.8), both in RGB and YUV colour spaces. The experiments show that the χ^2 -YUV detects the larger number of correct transitions but the histogram difference-YUV is the best choice in terms of overall performance (i.e. number of false alarms and correct detections). As a limitation we can note that the approach is not able to recognize the type of the gradual transitions. The main advantage of the clustering-based segmentation is that it is a generic technique that not only eliminates the need for threshold setting but also allows multiple features to be used simultaneously to improve the performance. For example, in their subsequent work Ferman and Tekalp [24] incorporate two features in the clustering method: histogram difference and pair-wise pixel comparison. It was found that when filtered these features supplement one another, which results in both high recall and precision. A technique for clustering based temporal segmentation on-the-fly was introduced as well.

4.2.7. Feature Based Temporal Video Segmentation

An interesting approach for temporal video segmentation based on features is described by Zabih, Miller and Mai [25]. It involves analyzing intensity edges between consecutive frames. During a cut or a dissolve, new intensity edges appear far from the locations of the old edges. Similarly, old edges disappear far from the location of new edges. Thus, by counting the entering and exiting edge pixels, cuts, fades and dissolves are detected and classified. To obtain better results in case of object and camera movements, an algorithm for motion compensation is also included. It first estimates the global motion between frames that is then used to align the frames before detecting entering and exiting edge pixels. However, this technique is not able to handle multiple rapidly moving objects. As the authors have pointed out, another weakness of the approach are the false positives due to the limitations of the edge detection method. In particular, rapid changes in the overall shot brightness, and very dark or very light frames, may cause false positives.

Although introducing a novel approach to temporal parsing, especially the detection of gradual changes, this algorithm does not bring any improvement regarding efficiency. It extracts edges from the uncompressed domain, and by that intensifies feature extraction so that the overall processing time increases.

4.2.8. Model Driven Temporal Video Segmentation

The video segmentation techniques presented so far are sometimes referred to as *data driven, bottom-up* approaches. They address the problem from data analysis point of view. It is also possible to apply *top-down* algorithms that are based on mathematical models of video data. Such approaches allow a systematic analysis of the problem and the use of several domain-specific constraints that might improve the efficiency.

Hampapur, Jain and Weymouth [26] present a shot boundary identification approach based on the mathematical model of the video production process. This model was used as a basis for the classification of the video edit types (cuts, fades, dissolves). For example, fades and dissolves are chromatic edits and can be modelled as:

$$S(x, y, t) = S_1(x, y, t) \cdot \left(1 - \frac{t}{l_1}\right) + S_2(x, y, t) \cdot \left(1 - \frac{t}{l_2}\right) \quad (4.16)$$

where $S_1(x, y, t)$ and $S_2(x, y, t)$ are two shots that are being edited, $S(x, y, t)$ is the edited shot and l_1, l_2 are the number of frames for each shot during the edit.

The taxonomy along with the models are then used to identify features that correspond to the different classes of shot boundaries. Finally, feature vectors are fed into a system for frames classification and temporal video segmentation. The approach is sensitive to camera and object motion.

Another model-based technique, called differential model of motion picture, is proposed by Aigrain and Joly [27]. It is based on the probabilistic distribution of differences in pixel values between two successive frames and combines the following factors:

- a small amplitude additive zero-centered Gaussian noise that models camera, film, digitizer and other noises;
- an intra shot change model for pixel change probability distribution resulting from object and camera motion, angle, focus and light change;

a shot transition model for the different types of abrupt and gradual transitions. The histogram of absolute values of pixel differences is computed and the number of pixels that change in value within a certain range determined by the models is counted. Then shot transitions are detected by examining the resulting integer sequences. Experiments show 94-100% accuracy for cuts and 80% for gradual transitions detection.

Yu, Bozdagi and Harrington [28] present an approach for gradual transitions detection based on a model of intensity changes during fade out, fade in and dissolve. At the first pass, cuts are detected using histogram comparison. The gradual transitions are then detected by examining the frames between the cuts using the proposed model of their characteristics. For example, it was found that the number of edge pixels have a local minimum during a gradual transition. However as this feature exhibits the same behaviour in case of zoom and pan, additional characteristics of the fades and dissolves need to be used for their detection. During a fade, the beginning and end image is a constant image, hence the number of edge pixels will be close to zero. Furthermore, the number of edge pixels gradually increases going away from the minimum in either side. In order to distinguish dissolves, the so called double chromatic difference curve is examined. It is based on the idea that the frames of a dissolve can be recovered using the beginning and end frames. The approach has low computational requirements but works under the assumption of small object movement.

Boreczky and Wilcox [29] use Hidden Markov Models (HMM) for temporal video segmentation. Separate states are used to model shot, cut, fade, dissolve, pan and zoom. The arcs between states model the allowable progressions of states. For example, from the shot state it is possible to go to any of the transition states, but from a transition state it is only possible to return to a shot state. Similarly, the pan and zoom states can only be reached from the shot state, since they are subsets of the shot. The arcs from a state to itself model the length of time the video is in that particular state. Three different types of features (image, audio and motion) are used:

- a standard grey-level histogram distance between two adjacent frames;
- an audio distance based on the acoustic difference in intervals just before and just after the frames and
- an estimate of object motion between the two frames.

public

The parameters of the HMM, namely the transition probabilities associated with the arcs and the probability distributions of the features associated with the states, are learned by training with the Baum-Welch algorithm. Training data consists of features vectors computed for a collection of video and labelled as one of the following classes: shot, cut, fade, dissolve, pan and zoom. Once the parameters are trained, segmenting the video is performed using the Viterbi algorithm, a standard technique for recognition in HMM.

Thus, thresholds are not required as the parameters are learned automatically. Another advantage of the approach is that HMM framework allows any number of features to be included in a feature vector. The algorithm was tested on different video databases and has been shown to improve the accuracy of the temporal video segmentation in comparison to the standard threshold-based approaches. Unlike the algorithms presented in previous sections, model driven algorithms for temporal video parsing tackled the problem of robustness and precision by applying more complex analysis of extracted feature set. Methods that model the way videos are being edited [26], [28] resulted in similar approaches that utilised compressed domain features. In addition, reported precision and recall in [29] are high and even show that the algorithm is very reliable and robust to camera and object motion.

5. Cross-media retrieval integration

5.1. The EASAIER system architecture

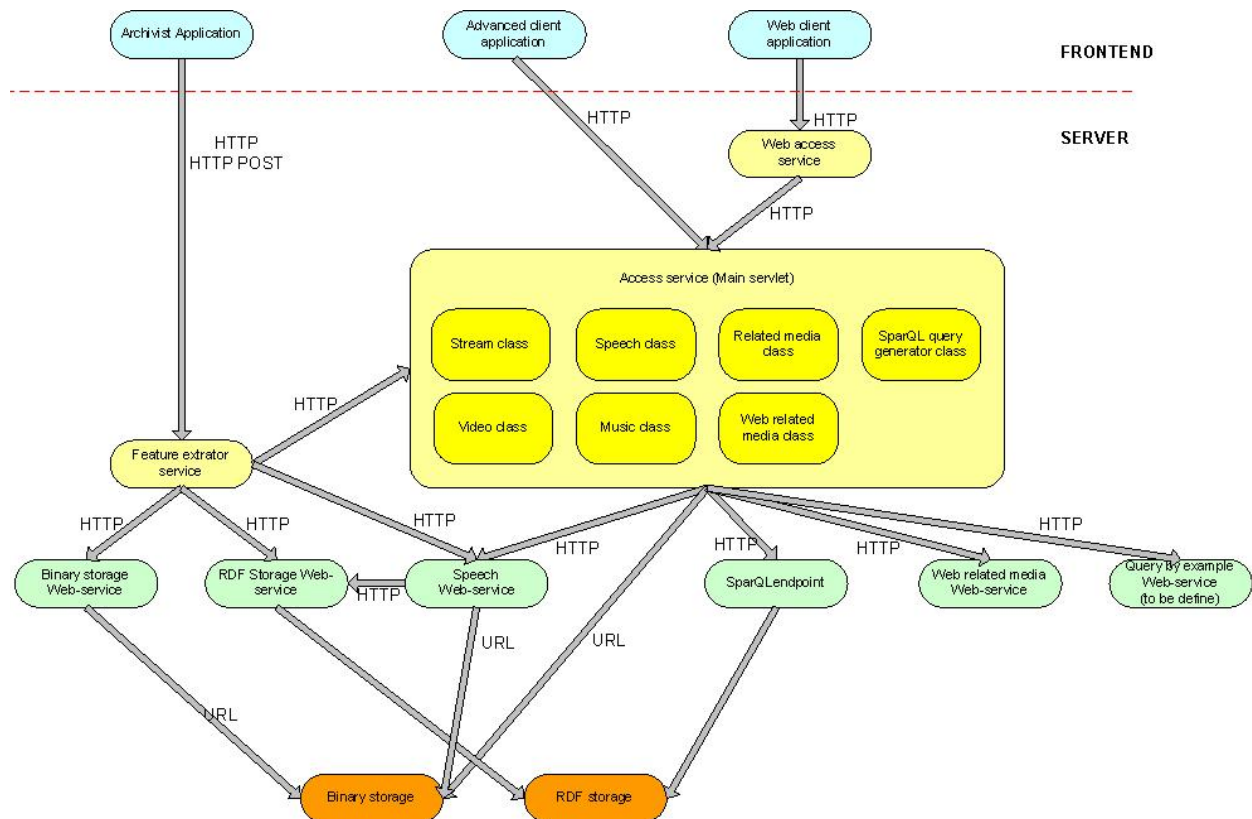


Figure 5.1. EASAIER global system architecture

Figure 5.1 shows the global system architecture of the EASAIER system. The cross-media functionality is integrated in the components “Related media class” and “Web related media class” in the main Access service. The “Related media class” uses the Sparql endpoint and rdf storage to retrieve related media in the current archive. It could be any information type, such as videos, biographies and photos. The “Web related media class” uses web-services available on the web for from services such as Google, YouTube, Yahoo, Dailymotion, Amazon...etc... The information retrieved from these web services is then re-formatted and merged to be embedded in the HTML pages of EASAIER system results.

The following sections will detail the workflows for the different cross-media retrieval scenarios.

5.2. Web-related media workflow

The web-related media retrieval functionality is only available in the web client application since the web client is more appropriate for this kind of search. The advanced client application, named Sound Access, is dedicated to retrieval and track manipulation in the archive, whereas the web client application is more devoted to quick search in the archive and cross-media retrieval.

Figure 5.2 presents a search done with the web client application. The user does a search according to different criteria: author, title or any other criteria available on the archive. Then, he can select a specific track and at this time a “web related media search” will be done automatically.

The workflow is depicted in Figure 5.3.

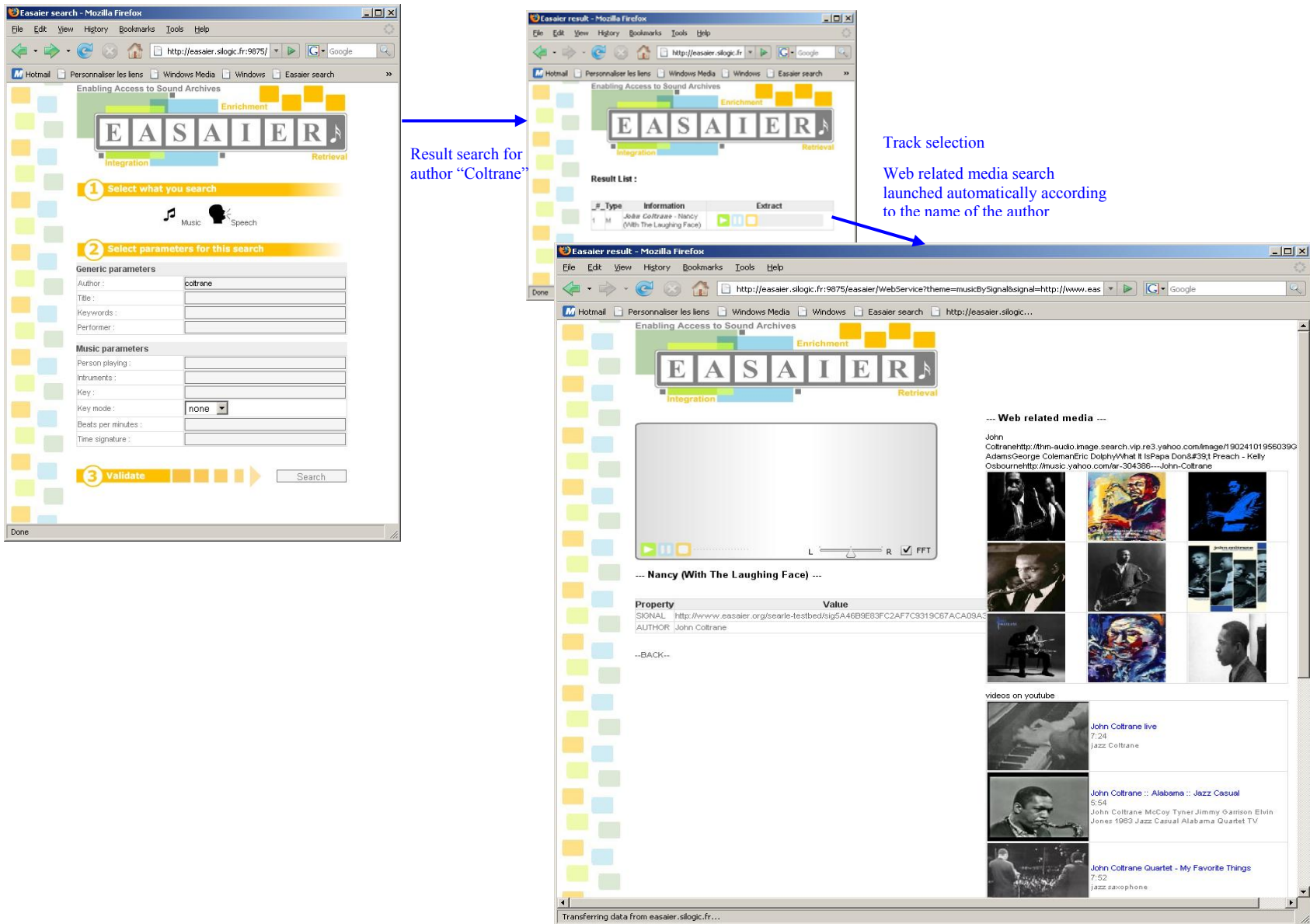


Figure 5.2. Search for author “Coltrane” in the Web client application. The web related media search is launched automatically after the selection of a track.

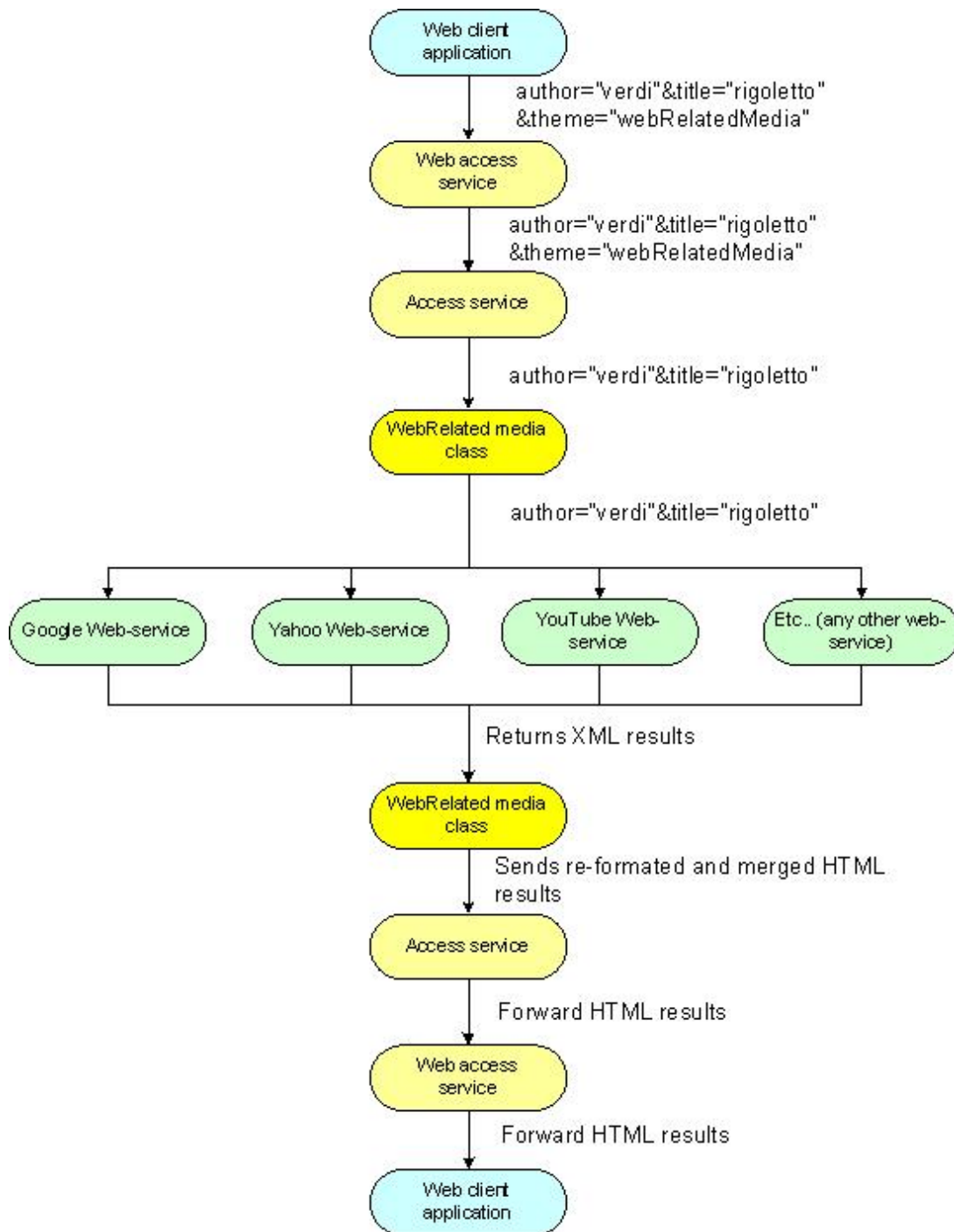


Figure 5.3. Web Related media workflow

Figure 5.3 presents the workflow for the web related media. The author as selected a specific track (author="verdi"&title="rigoletto") and corresponding web related media will be recovered automatically by the following way.

The web access service redirects the query to the main access service. The web related media class will then interrogate different web services available on the world wide web (Google, youtube, Dailymotion, Amazon...). The different results are then merged and re-formatted to be embedded in an HTML pages. The reformatted result is then forwarded until the web client application to be displayed.

5.3. Web-related content retrieval integration

The web-related media component has been integrated as part of the web client application. Once a search performed using the web client application is successful (i.e. one or more results are yielded from the querying the archive), each audio asset can be viewed in detail, with all the metadata information, streaming audio or video etc. Alongside, the web-related media web service delivers additional content in the form of the assembled results that were retrieved by posing the metadata information to a set of predefined external content providers.

While the related media about the archive content's assets stored by the EASAIER system is static (unless manual metadata import takes place), the related media retrieved from the web may be different each time a query about a specific resource is invoked.

For the majority of the external content providers, the REST [9] protocol is used to send a request and retrieve the results in an XML format that adheres to a certain predefined structure. Those XML results include URIs to multimedia resources on the web, such as images and video and also any associated metadata, such as tagging information, a textual description or for example the duration of a videoclip. Applying XSL stylesheets on the XML results, we are able to display the results in a customized fashion, choosing the type and amount of metadata we wish to display, the layout, size of embedded content such as images, etc.

In the following, we describe this workflow in a brief example that showcases the communication between the EASAIER web client and the YouTube service.

Once a user selects an audio asset from the result list in the EASAIER web client, the Web Related Media component uses the result's metadata, such as the artist name, as the query parameter to the external content providers. In the case of YouTube, the following message is propagated to the YouTube REST API:

```
http://www.youtube.com/api2_rest?method=youtube.videos.list_by_tag&dev_id=l2xAWILQ19A&tag=john%20coltrane&page=1&per_page=3
```

The method *youtube.videos.list_by_tag* invokes the YouTube internal search engine to search for videos that are associated with the specified tag keyword using the YouTube "by-relevance" ranking. The YouTube service uses its internal search engine, retrieving the most relevant results, and sends back a list of results and metadata for each result in XML format:

```
<?xml version="1.0" encoding="utf-8"?>
<ut_response status="ok">
  <video_list>
    <total>1725</total>
    <video>
      <author>gsmaior</author>
      <id>q6WwuxqXPOg</id>
      <title>John Coltrane live</title>
      <length_seconds>444</length_seconds>
      <tags>jazz Coltrane</tags>
      <url>http://www.youtube.com/?v=q6WwuxqXPOg</url>
      <thumbnail_url>http://s2.ytimg.com/vi/q6WwuxqXPOg/default.jpg</thumbnail_url>
      (...)
    </video>
    (...)
  </ut_response>
```

For the web related media service, this XML document is subsequently processed using a custom XSL stylesheet, generating the desired HTML markup. The following excerpt of the corresponding XSL document gives an example of how to format the relevant metadata entries inside an HTML table:

```

<table>
  <xsl:for-each select="video_list/video">
    <xsl:variable name="seconds"><xsl:value-of select="length_seconds"/></xsl:variable>
    <tr>
      <td>
        <a href="{url}"></a>
      </td>
      <td>
        <div>
          <a href="{url}" style="color:#00f"><xsl:value-of select="title"/></a><br/>
          <span><xsl:value-of select="floor($seconds div 60)"/>:<xsl:value-of select="$seconds mod
60"/></span><br/>
          <span style="font-size:10"><xsl:value-of select="tags"/></span>
        </div>
      </td>
    </tr>
  </xsl:for-each>
</table>

```

After applying the XSL stylesheet to the XML source, the resulting table element as part of the HTML output looks as follows:

```

(...)
<table>
  <tr>
    <td>
      <a href="http://www.youtube.com/?v=q6WwuxqXPOg"></a>
    </td>
    <td>
      <div>
        <a href="http://www.youtube.com/?v=q6WwuxqXPOg" style="color:#00f">John Coltrane
live</a><br/>
        <span>7:24</span><br/>
        <span style="font-size:10">jazz Coltrane</span>
      </div>
    </td>
  </tr>
  (...)
</table>
  (...)

```

The representation of the HTML output can be customized further by associating a CSS template.

6. Web related content retrieval components

Google

Google provides the AJAX Search API¹ that allows the integration of their powerful search mechanism in the form of an embeddable widget on a webpage. Upon browsing a result from the EASAIER Web based client, the Web Related Media component is invoked, and the metadata from the query result is posed as an initial query term to the Google *pre-search* functionality of the embedded widget. When the page is loaded, the results from the Google web, images, video and blog search are retrieved and displayed as part of the result page. The display and layout properties of the Google search widget can be configured using Javascript. The actual communication between the widget and the Google server happens through an AJAX component which relies on Javascript code that is hosted remotely, i.e. by Google. Thus, any changes to the API won't mandate local updates. The AJAX search widget takes a different approach than relying on sending REST or SOAP based request and receiving and processing XML or JSON result sets. Using e.g. REST and XML allows for more freedom for the developer to assemble the search results, by operating on raw result data (image or video URLs and metadata for example). Google has recently released the GData² standard to communicate with the Google APIs as an alternative. GData relies on REST, and it is heavily based on Atom and RSS technologies.

YouTube

YouTube is a platform that allows users to view and upload videos. Videos are associated with textual descriptions, ratings, and tagging metadata that is created by the users. It is possible to conduct a keyword search by rating, view count, age and relevance of the video. The query result ranking by relevance uses the internal YouTube search engine algorithm. YouTube provides a public API³ that allows users to conduct a search, and it is also possible to embed the videos hosted by YouTube remotely inside a webpage. The YouTube search API currently used by the Web Related Media component relies on HTTP REST, and query results are returned in XML. When the Web Related Media component poses a query to YouTube, it specifies the query keywords, the number of desired results and that the results should be ordered using the “by-relevance” ranking. The results are sent back in XML and are then parsed by the Web Related Media component using XSL. For each result item the video thumbnail and basic metadata such as the video's title, duration, etc. are displayed. Clicking on the thumbnail of an item navigates the user to the YouTube page of the video. Recently, YouTube has launched their new API YouTube GData API⁴, whereas the REST API will remain being supported.

Yahoo Images & Music

Yahoo provides a web search service⁵ for the easy integration of Yahoo search results inside a web page. We use the Yahoo Images and Music search services for retrieving relevant images found by Yahoo on the web and related artists, respectively. The API relies on REST and query results are sent to the requesting party in XML. Results are parsed by Web Related Media

1 <http://code.google.com/apis/ajaxsearch/>

2 <http://code.google.com/apis/gdata/>

3 <http://www.youtube.com/dev>

4 <http://code.google.com/apis/youtube/overview.html>

5 <http://developer.yahoo.com/search/>

public

component using XSL, and the images and information about related artists are displayed as part of the overall search results.

Amazon

The e-Commerce platform Amazon.com offers retail sales of a large variety of products. Amazon Web Services provides an API⁶ with a large number of functionalities for websites and client-side application. This includes the Amazon E-Commerce Service that allows product search to retrieve information about items from the Amazon marketplace, such as images, reviews, etc. The service can be accessed over HTTP using either REST or SOAP. The Web Related Media component uses the REST based approach: a keyword query is posed to the service, and additional parameters are set to make the query specify the type of results, including the category to be searched which is set to “Music”. Results are transmitted in XML format and parsed by the Web Related Media component using a predefined XSL document. The final output is the Amazon.com web widget that contains informal product information, a thumbnail image of the item and an external link to the Amazon web page of the item for additional information.

eBay

The online auction platform eBay offers an API for developers through the eBay Developers Program⁷ for the implementation of applications that integrate with eBay. The services are exposed over standardized protocols, including REST and SOAP. The Web Related Media component uses the REST based approach to send queries; results are sent back in XML. The result consists of the resulting top relevant items, according to eBay’s native search engine, like CDs, posters, collector items etc. that are related to the resource. The XML result is parsed using XSL, and then embedded in the eBay widget with the image of each item along with textual descriptions and external links to the item’s eBay web page.

6 <http://developer.amazonwebservices.com>

7 <http://developer.ebay.com/common/api/>

7. Related media in the archive

7.1. Related media workflow

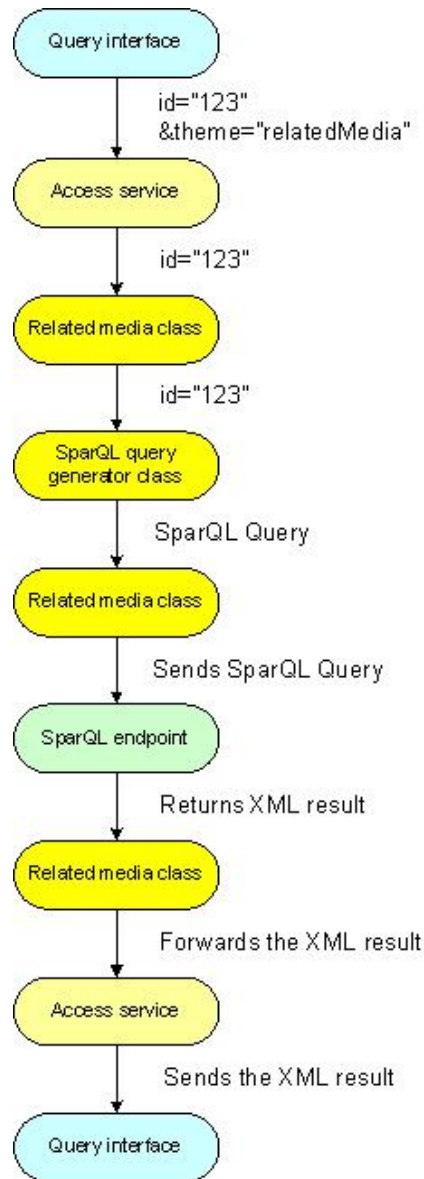


Figure 7.1. Related media workflow

The related media workflow runs automatically when a specific track is selected in the query result tab of the EASAIER sound-access client application. The specific track has id="123" which correspond to the signal URI in the rdf storage. The main access service and specially the related media class calls the SPARQL query generator class to get the correct query to interrogate the rdf storage. The SPARQL query for related media is constructed according to the database ontology. It can retrieve pictures of the authors, bibliography and any other information associated to the track. The results are then send back to the client application for display.

7.2. Integration in the Sound access client application

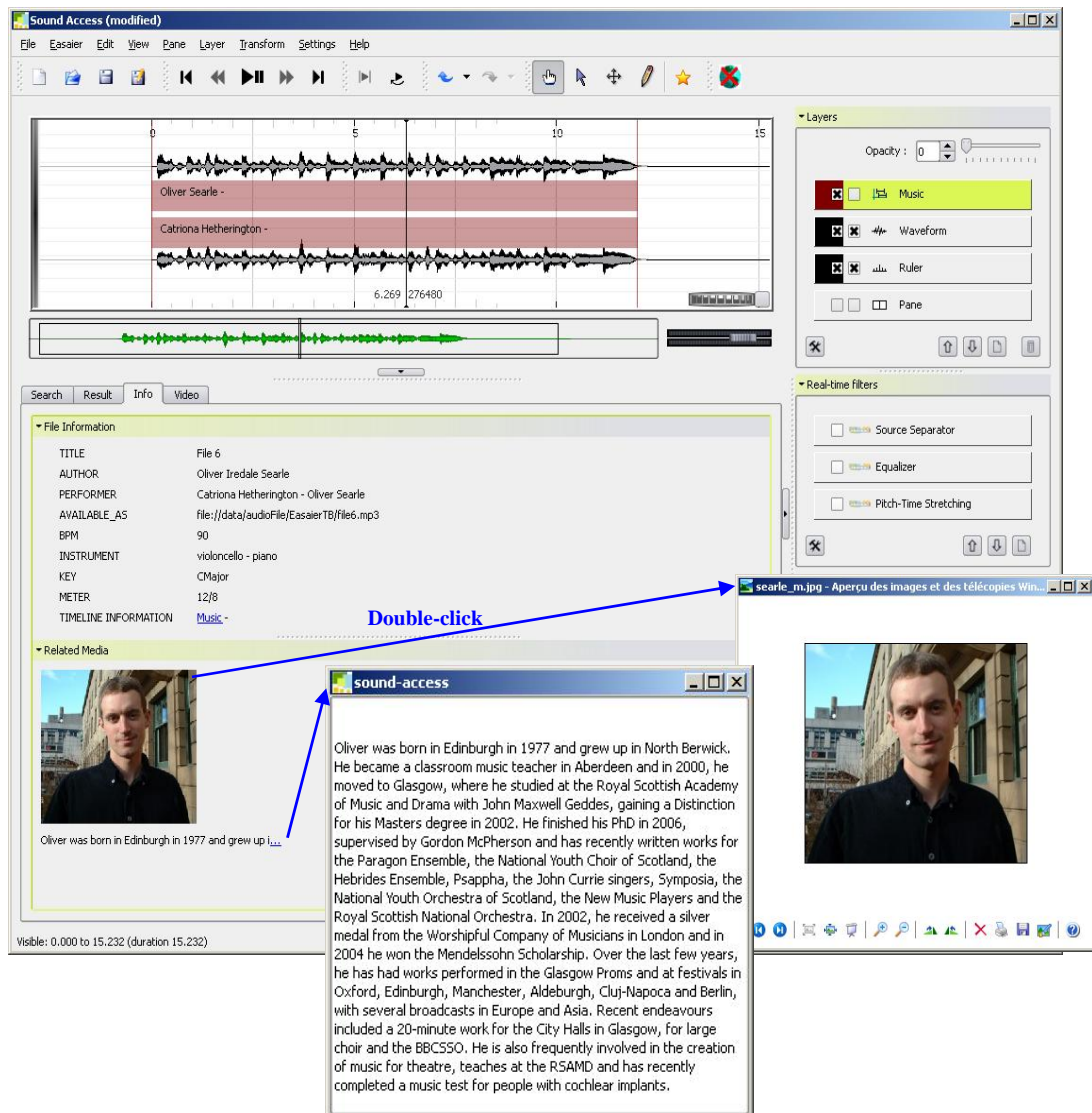


Figure 7.2. Related media in the archive

Figure 7.2 shows the way the related media retrieved from the archive are displayed. They are retrieved according to a track result. It could represent any information related to the author (bibliography, pictures...), the album (cover...), the current track (score, lyrics...) and more. All related media retrieved and sent by the server will be displayed in the client application.

The related media can be opened on the client machine as long as the client has the correct application to read the content. So video, pictures, textual information will be supported by most client machines. If specific formats are used by an archive for related media they will be able to use the EASAIER system without any further development. For example if scores are stored in a specific format, the end user will be able to retrieve it with the Sound Access client and open it with an existing application.

8. Video analysis using spectral clustering to support cross-media queries

Shot and scene boundary detection can be seen as essential steps of video analysis and cross-media retrieval. Conventional shot detection methods analyze consecutive frame similarities therefore most of the general information about the shot is lost. Our work is motivated by the assumption that a shot boundary is a global feature of the shot rather than local. General knowledge about the shot is cumulated during the time from the information included in every frame. Information about the boundary is extracted indirectly from the information about the interaction between two consecutive shots. Spectral clustering keeps each frame's contribution in the objective function. By optimizing the objective function when clustering frames, individual shots are separated, and cluster bounds are used for detecting shot boundaries. Using all available data describing the shot, improves the performance of the boundary detection algorithm. By keeping the information in the objective function every frame of the shot has its contribution to the overall knowledge about the boundary. As frames that belong to the same shot are temporally aligned, cluster borders will be points on the time axis. In our algorithm spectral clustering algorithm Normalized Cut [30] is used for clustering. Specially created heuristics is applied to analyze results of the clustering and give final results. Clustering the whole video based on the visual similarity of frames exhibit another interesting property. Resulting clusters will have visual similar frames no matter if they are temporally adjacent or not. This feature can be used to automatically create video summaries.

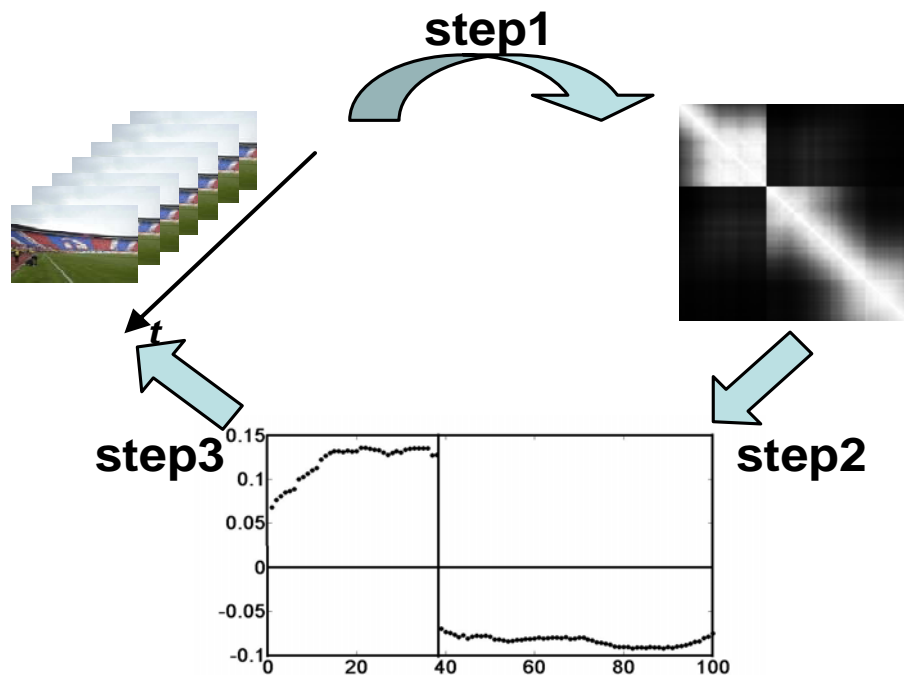


Figure 8.1. Diagram of the spectral clustering algorithm. Step1: pair wise similarity measure is used to map the underlying structure of the dataset to the similarity matrix. Step2: second smallest eigenvector with nearly piece wise constant values reveals the block based structure of the similarity matrix. Step3: information from the eigenvector is used to bipartition the dataset.

8.1. Spectral clustering for shot boundary detection

As shown in figure 8.1, creation of the similarity matrix is the starting point of every spectral clustering algorithm. MPEG-7 colour layout descriptor is used for the low level representation frames. This descriptor is obtained by applying the DCT transformation on a 2-D array of local representative colours in luminescence Y and chrominance C_r and C_b planes [31]. Every frame is represented by feature vector F_i with 58 entries. Similarity between frames is calculated by:

$$w_{ij} = e^{-\frac{d(F_i, F_j)}{\sigma^2}} \quad (8.1)$$

where $d(F_i, F_j)$ is the distance between two vectors as defined in [31]. Calculating frame by frame, similarities for the whole video would take $O(N^2)$ operations, where N is the number of frames. The number of operations can be reduced assuming that frames that are far away in the temporal axis do not belong to the same shot. Specifically, we are using sliding window of 100 frames ($100 \ll N$) on which clustering is carried through. By using the sliding window number of operations needed to analyze whole video is reduced from $O(N^2)$ to $O(N)$.

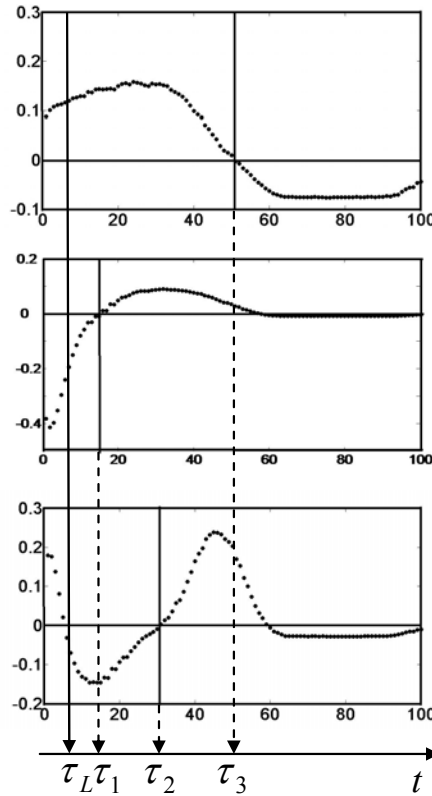


Figure 8.2. Similarity matrix eigenvectors used for finding the shot boundary candidates. Vertical axes correspond to eigenvector entries, showing each frame position relative to the boundary while each frame corresponds to a point on the horizontal axis. Possible boundaries are points on the time axis, where the sign of the eigenvector is changed for the first time. τ_L is the starting point of the analysis. τ_1, τ_2 and τ_3 are candidates for shot boundaries.

Three eigenvectors corresponding to the second, third and fourth smallest eigenvalues are used for describing the structure of video inside the window, Figure 8.2. Every eigenvector indicate possible choice of the shot boundary: τ_1, τ_2 and τ_3 . Candidates are chosen to be points on the time axis where values of the eigenvectors change sign for the first time, Figure 8.2. The point τ_L is the

public

starting point where the search for candidates starts, and is the minimal number of frames one shot can have. Constraint on the number of frames in a shot is necessary as the quality of spectral clustering depends on the size of clusters. [32] showed that if the similarity matrix is block stochastic spectral clustering will give the perfect clustering, or in our case will find shot boundaries [33]. In the simplest case, for cuts, the similarity matrix fulfil this condition so eigenvectors will be piece wise constant. Consequently position of the shot boundary would exactly match the position smallest shot candidate, τ_i . In the case of the gradual change, the similarity matrix will have a structure that is more or less similar to the block stochastic matrix, depending on the level of change in frames. In order to detect and classify these changes properly further analysis of eigenvectors is necessary. Lets define thresholds $NCut_{L1}$, $NCut_{L2}$ and $NCut_{L3}$ for normalised cuts in τ_1 , τ_2 and τ_3 respectively. Similarly, threshold values for the time difference τ_{Li} of the candidates depends on the actual distance between possible boundaries:

$$\tau_{Li} = g(\tau_{i+1} - \tau_i), i = 1, 2 \quad (8.2)$$

The reasoning algorithm for the analysis of eigenvectors is then as follows:

Input: Eigenvectors cut points τ_1 , τ_2 and τ_3 with respective $NCut$ values in these points.

Output: Single frame candidate for shot boundary

1. Initialize frame candidate: $\tau_B = \tau_L$
2. **IF** $NCut(\tau_1) < NCut_{L1}$ **THEN**
3. **IF** $(\tau_1 - \tau_2) > \tau_{L1}$ **THEN** $\tau_B = \tau_1$; **BREAK**;
4. **ELSE IF** $NCut(\tau_1) < NCut(\tau_2)$ **THEN** $\tau_B = \tau_1$; **BREAK**;
5. **IF** $NCut(\tau_2) < NCut_{L2}$ **THEN**
6. **IF** $(\tau_2 - \tau_3) > \tau_{L2}$ **THEN** $\tau_B = \tau_2$; **BREAK**;
7. **ELSE IF** $NCut(\tau_2) < NCut(\tau_3)$ **THEN** $\tau_B = \tau_2$; **BREAK**;
8. **IF** $NCut(\tau_3) < NCut_{L3}$ **THEN** $\tau_B = \tau_3$; **BREAK**;
9. **ELSE** there is no boundary in the window

Once the shot boundary candidate is found, statistical analysis of the $NCut$ value in the adjacent frames is done. First, $NCut$ value is calculated for 5 frames before and after the τ_B . Mean value μ_n and standard deviation σ_n of $NCut$ values are found. In [34] is showed that the performance of shot detection methods can be improved by the use of a threshold that adapts itself to the sequence statistics. Adaptive threshold used in our algorithm is defined by:

$$m_T = T_P \cdot \mu_N + T_D \cdot \sqrt{\sigma_N} \quad (8.3)$$

where T_P and T_D are experimentally evaluated.

An adaptive threshold algorithm is applied to the objective function values. If the objective function in point τ_B is less than the threshold defined in (8.3), τ_B is the shot boundary. If the objective function is greater than the threshold, current window contain no shot boundaries. By sliding the window over the time axis, whole video is segmented, and shot boundaries are extracted independently of the category of the change.

8.2. Shot boundary detection and key frame extraction using recursive Normalized cuts

Another spectral clustering approach to the shot boundary detection problem is to apply recursive clustering of frames within the sliding window until all borders are found. The motivation for this is to avoid dependencies between video structure and strict rules defined in the previous

public

subsection. The first step in the algorithm is the same as in section 8.1; a similarity matrix of the sliding window is created. Recursive clustering is applied to all frames within the window. After each clustering step as described in recursive clustering, the $NCut$ value is compared to the experimentally determined threshold $NCut_{recSD}$ and the decision is made if cluster is found or if it should be continued.

There are a few possible cases when clustering frames of the sliding window. The simplest case is when there is no shot change in the window. This is indicated with a high $NCut$ value in the first step. If this is the case, sliding window is moved so the last frame of the window becomes first frame of the newly positioned window. The second possible case is when there is one shot boundary within the window. This case is also simple to resolve, since $NCut$ value of first clustering step is significantly lower than $NCut$ value of the following steps, and at the same time it is lower than threshold used for clusters detection. Frames belonging to different shots will be clustered to distinct clusters, and the shot boundary is found as the point in time where clusters meet, as in Figure 8.3. The case when there are more than two shots in the sliding window results in a number of clusters distributed on the time axis. Shot boundaries are again found by looking for clusters borders on the time axis in every clustering step until all clusters are found. It is possible that one cluster contains frames that are separated in time. Even though these frames are clustered together they will still be detected as distinct shots since every cluster change correspond to the shot change Figure 8.3.

After the video is segmented into shots, we propose a new approach to estimate level of motion activity and extract key frame representative of the shot. We use information from the eigenvectors of the similarity matrix created from frames of each shot, and predefined reference shot to estimate level of motion activity without use of motion vectors. Together with motion activity estimation, information from eigenvectors is also used to find most representative frame of each shot. The leading assumption for our approach is that eigenvector structure can be seen as proper indicator of a dataset structure. By eigenvector structure we basically mean how close is the eigenvector to the ideal case, with all entries corresponding to a single cluster being equal. We define a measure for quantifying proximity of the real eigenvector to the ideal case and use it as a motion activity level indicator. In [32], it is shown that eigenvectors entries can be seen in terms of probabilities of Markov walk over the set of data points. It is also shown that if the stochastic matrix of the Markov random walk is block stochastic eigenvectors will be piecewise constant.

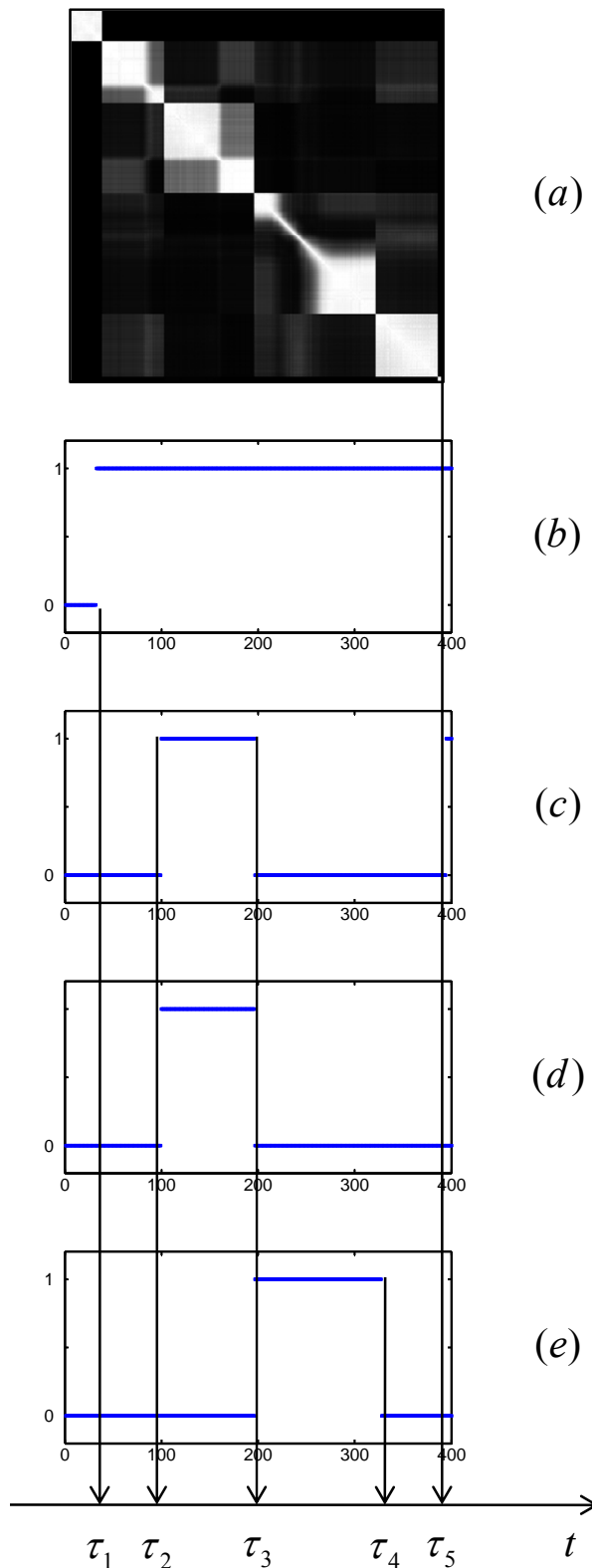


Figure 8.3. Illustration of the recursive shot detection algorithm. (a) Pair wise similarity matrix created from the frames in the sliding window. (b)-(e) Clusters of different recursive clustering steps. Each point on the horizontal axis corresponds to one frame of the sliding window. For each frame value of 1 or 0 is assigned based on the clustering result of each step. 5 shot boundaries are found by detecting borders between on the time axis clusters in each step.

public

This means that when in the step t and the walk is in state i , the probability $p_{i,j}$ that in the next step $t+1$ the walk will move to any state j with $p_{i,j} > 0$ is constant and is equal to the stationary distribution of node j , π_j . The structure of the eigenvector can be seen in terms of Markov random walks. We assume that the more one cluster is away from the ideal case, the more its eigenvalues are far from the piecewise constant case. When moving away from the ideal, elements of the cluster are connected with similarities that are not constant anymore and lay in some range. The more the cluster is away from the ideal case, the bigger the range is. In the transition probability perspective this means that probabilities from going to different points in the cluster is no longer constant, resulting in a distribution which is also not constant for the members of the same cluster.

In order to be able to make use of the eigenvector structure we define a measure $Mov(C)$ for describing how much is a cluster C with n_C frames away from its ideal case. Ideal case is represented with mean value of eigenvector entries defined in equation (8.3). For every video we use the most static shot ref with n_{ref} frames as reference cluster. To calculate $Mov(C)$, we apply $NCut$ algorithm to the set of frames containing frames of the shot that we are analyzing on one side and set of frames from the reference shot on the other side Figure 8.4. We calculate the mean value of eigenvector entries belonging to both clusters:

$$x_{mean}^{(2)}(C) = \frac{1}{n_C} \sum_{i \in C} x^{(2)}(i) \quad (8.3)$$

and

$$x_{ref}^{(2)}(C) = \frac{1}{n_{ref}} \sum_{i \in ref} x^{(2)}(i) \quad (8.4)$$

$Mov(C)$ is then calculated using following equation:

$$Mov(C) = \frac{n_{ref} \sum_{i \in C} |x^{(2)}(i) - x_{mean}^{(2)}(C)|}{n_C \sum_{i \in ref} |x^{(2)}(i) - x_{ref}^{(2)}(C)|} \quad (8.5)$$

Motion activity measure defined in this way is ratio of average difference between eigenvector entries and mean eigenvector value of cluster C and referent cluster ref . Level of motion activity in the shot is then described with defined measure $Mov(C)$. It is also possible to estimate type of the change in the shot by analyzing the temporal structure of eigenvector entries. Different types of motion activity will have different structure of the change on the time axis. For example, a shot with global motion will have frames that are oscillating over the mean value, as shown in Figure 8.4. On the other hand, shots with moving objects or with zooming effects will have eigenvector entries whose values have tendency to grow bigger or to go lower over time, also as shown in Figure 8.4. Since zooming is done smoothly, eigenvector values are also smoothly decreasing. Key frame extraction is done using the same principles as motion activity estimation. Key frame i of a shot S should satisfy following condition:

$$\sum_{j \in S} w_{i,j} = \max_{n \in S} \left\{ \sum_{m \in S} w_{m,n} \right\} \quad (8.6)$$

This means that the key frame will have the largest total similarity to all frames of the shot S . Using eigenvectors frame i is found as the frame which second eigenvector entry $x^{(2)}(i)$ having the largest absolute difference from zero.

Figure 8.5 shows three typical cases from our experiments. Example of an abrupt change of the shot is shown in the row I. Clear block structure is defined in the similarity matrix figure 20 (a), and all three eigenvectors have the cut in the same frame, Figure 8.5 (b-d). The case of a gradual change is shown in Figure 8.5, row II. Block structure of the similarity matrix is not so clear in this case which results in three different cuts in each eigenvector. To detect the proper position of the shot change algorithm for eigenvectors analysis is employed. Finally the case where there is no boundary in the window is shown in Figure 8.5, row III. All frames within the window have high similarities, and resulting eigenvectors have unclear structure.

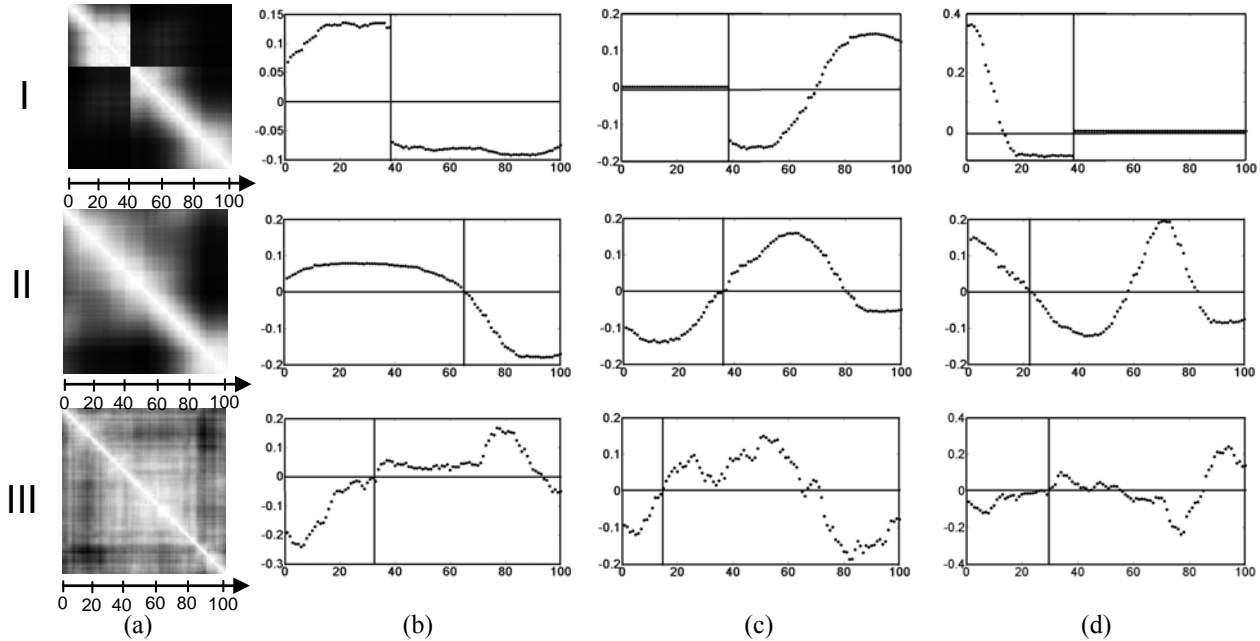


Figure 8.5. Three typical examples found in Multiway cuts experiments. (a) Similarity matrix. (b)-(d) Second, third and fourth eigenvectors providing shot boundary candidates. Horizontal axis corresponds to single frames (1-100) inside the sliding window. Respective eigenvector values are shown on the vertical axis. First sign change in the eigenvector values indicates the shot boundary candidate. Row I: Abrupt change of shots, similarity matrix has the clear block structure. All eigenvectors give the same frame for cut candidate. Row II: Gradual change of shots, block structure of the similarity matrix is not so clear like in the cut case. Every eigenvector gives different candidate for shot boundary. Row III: No shot change within the sliding window. The similarity matrix consists of frame similarities from the same shot.

Table 8.1. Boundary detection results for three different genres of videos.

Method	Video type	TP	FP	FN	Recall[%]	Precision[%]
Multiway cuts	News	312	63	76	80	83
	Documentary	254	61	78	78	80
	Music	306	54	82	76	78
Recursive Ncut	News	350	32	38	90	91.6
	Documentary	302	42	30	90.9	87
	Music	332	45	66	88	83.4

Table 8.2. Boundary detection results for three different genres of videos

Method	Data	Recall[%]	Precision[%]
Multiway cuts	TrecVid2007	72	53
Recursive	TrecVid2007	81	73

For showing the actual performance of the algorithm, standard precision and recall are used. Results obtained with three different video types are shown in Table 8.1. TP refers to the number of correctly detected boundaries. FP is the number of boundaries incorrectly detected, while FN is the number of missed boundaries. Generally recursive shot boundary detector showed better results, with precision and recall values around 90 percent. Multiway boundary detection showed to be less stable to content change since set of rules defined for boundary detection put too many constraints. On the other side recursive *NCut* is completely independent on the dataset since *NCut* threshold is only value which depends on actual data, that algorithm uses to detect boundaries.

Boundary detection for music videos was a difficult task. Total editing freedom makes it hard to use some predefined reasoning for eigenvectors analysis. A high number of false positive results was due to big changes in scenes like change in the light, overall colour of the scene and extensive use of animation effects. This resulted in a change of the colour appearance of frames, which is reflected in big distance between feature vectors. However, even with many factors that make shot boundary task difficult for music videos, the results were reasonable.

In this chapter we introduced a shot detection method that segments the video by using spectral clustering technique. We tested out algorithms with TrecVid2007 dataset. We participated in TrecVid shot boundary task with Multiway cuts algorithm. To compare both algorithms we also ran the recursive *NCut* boundary detector on the selected set of TrecVid videos. We present TrecVid results in the Table 8.2. It is described how objective function, optimized by spectral clustering, can save the contribution of each frame to the overall structure of the shot. Performance of the algorithm for spectral clustering of frames into shots together with applied heuristics for eigenvectors analysis is tested on three different video genres. It is shown that performance of algorithm is highly dependent on the choice of the similarity measure. We are investigating ways to improve the similarity measure between frames. By making the similarity matrix less sensitive to changes in the video clustering results will expose the video structure more reliable. We also showed that by extending the domain of clustering from frames to shots, it is possible to cluster shots with similar content into same cluster. This property of our technique can be extended to enable automatic creation of video summaries

9. User Evaluation

This section discusses feedback from the student evaluators provided in their written evaluations of the client from Apr 08, focussing on their comments relating to performance, usefulness and usability of the cross-media retrieval system. It should be noted that they were testing an Alpha release of the client and, as such, were unable to examine some aspects such as precision and recall, as they were only using one query. However their comments are enlightening and offer some useful guidance about their contexts and uses which should inform development of this aspect of the EASAIER system.

All of the students, except one who made no comment, were supportive of the system offering a range of types of media to the user based on an initial search query:

'I would like to see videos relating to performer, composer or style of music connected with a music file, giving the user option to browse through this type of media, which would be another attracting feature of the EASAIER program.' (002)

'I thought that it was very smart how when you picked your piece of music that in the information section it loaded up related pictures to your music. I thought that it might be even more helpful if it brought up related videos, possibly of people playing the piece' (005).

'I noticed the related media section as being very useful with images and such like that connects with the entered search' (010)

A range of media were proposed which they felt would be useful to them as practicing (mainly) classical musicians. These are listed below in full. Some are derived from the evaluator's comparisons with other services they were familiar with, such as Naxos Music Library and Grove Music Online.

Text:

- Pronunciation guide of composers and performers
- List of composers with biographies and discographies
- 'Junior section' which contains the story of classical music, music for kids, meet the instruments of the orchestra
- Glossary
- Sound files
- Images
- Historical information
- Top songs
- Other users purchases
- Scores (notated music)
- Biography of composer and some dates (composer's dates, premier dates)
- Opera synopsis and libretti
- Biographies of the artists and composers
- Articles
- Sleeve notes

Audio:

- History of Opera
- Pronunciation guide
- Glossary
- Fundamental terms
- Podcasts

- Text searches
- Albums
- Several different recordings of same work
- Audio commentary by professional
- Gathering many interpretations of a piece
- Original version of a transcribed piece

Video:

- Videos relating to performer, composer or style of music
- Videos (with pause and rewind)
- Videos of people playing the piece

Images:

- Visual photographs
- Front covers of the music
- CD covers
- Score covers
- Related art
- Images

These students generally state they would use EASAIER to help them practice and improve their performance skills, rather than analyse content. There was a general call for a wide range of multimedia to be offered in search results, including images, video, biographies, libretti, lyrics, notated scores:

If, for example, I was playing a transcription of a Bach cello suite, I would be able to search for a recording (of both the transcription and the original depending how wide I make my search), then in the related media tab I would be able to browse articles, CD covers, sleeve notes, available score covers, related art (for example a Debussy search may yield a Turner painting), all of which are useful not only for studying my instrument and repertoire but also for writing general history of music essays for coursework. (009)

Comparisons were made with Naxos frequently, which they feel has wide coverage, and they wanted to have confidence they had a similar choice of material which they could then narrow down by gradually refining their search using the search parameters. These comments should be taken in the context of their evaluations. During the testing of the client it was only possible to retrieve and interact with one piece of music. This limited the ability of the students to test the range of materials available from the system, so their comments are based on a hypothetical stance, although the range of media offered by the system (audio, image, video, text) are representative of the software's functionality.

9.1. Discussion

These users are enthusiastic about the potential of integrated software such as EASAIER. Their evaluations applaud the ability of the package to offer links between media, saving them the trouble of searching elsewhere. They are therefore keen that a search will offer a range of material that reflects their needs:

'if the program had a function to download a score and to let the musician follow it whilst the music was being played, this would be the most helpful thing in the world.' (004)

public

'an extra part of the system which would make it an invaluable resource would be the addition of scores. The scores would be on a tab like the video tool and would scroll along with the music so that it would be possible to see how the piece was written as well as what it sounds like.' (003)

This presents some difficulties. While it is possible the collection they are searching does meet the user's criteria for relevant surrounding material, it is equally possible this is either not accessible for some reason, technical or legal, or does not exist. It is important, therefore, to educate the user about the range of choice available for multimedia and give them the option to expand or contract the search to include or exclude certain items or types of material. This should be flexible to realistically represent the content of the collection. The range of material requested by the students (above) indicates a need for careful indexing and additional generation of bespoke items specifically for the EASAIER user, such as commentaries.

'Unlike EASAIER, Naxos also contains audio tools for the user, which comprise of audio transcripts for example the History of Opera, Pronunciation Guide, Glossary, Fundamental Terms, Podcasts and Text searches. I think EASAIER could be improved if it also incorporated some audio tools to give different mediums to the user.' (001)

Scores were mentioned frequently and a score-following functionality would certainly enhance these users' experience. Although these users focus on Western art music, referring to opera synopses and libretti, for example, these comments are easily transposed to other genre users, such as pop or folk musicians who could be offered lyrics and performer histories, for example.

9.2. Summary and conclusion

We have shown how this group of user evaluators are keen to be offered a range of multimedia related to their initial query. Amongst the key media they request is:

- A range of audio examples of performances of the same work,
- notated music which they can follow while listening to a piece,
- video of performances which they can interact with,
- backup historical and contextual text material
- backup historical and contextual visual material
- backup historical and contextual audio material

It is recommended that the search system will allow the user to refine the query to reflect their contextual needs at the time of the search, within the constraints of the collection(s) being used.

10. References

- [1] Barrett, S., Duffy, C., and Marshalsay, K., "HOTBED (Handing On Tradition By Electronic Dissemination)," Royal Scottish Academy of Music and Drama, Glasgow, Report March 2004. www.hotbed.ac.uk
- [2] M. Asensio, "JISC User Requirement Study for a Moving Pictures and Sound Portal," The Joint Information Systems Committee, Final Report November 2003. www.jisc.ac.uk/index.cfm?name=project_study_picsounds
- [3] Tsakonas, G., and Papatheodorou, C. Exploring usefulness and usability in the evaluation of open access digital libraries. *Information Processing & Management*, 44(3), pp 1234-1250, 2008.
- [4] Inskip, C., and Barrett, S., "Users' Evaluations Report," Internal EASAIER report May 2008.
- [5] Barrett, S., and Inskip, C., "Using users' evaluations to determine context," *Information Interaction in Context BCS symposium*, London, October 2008.
- [6] S. Kami Makki, Jaina Sangtani, "Data Mashups & Their Applications in Enterprises," *ICIW Third International Conference on Internet and Web Applications and Services*, pp. 445-450, 2008.
- [7] Urs Gasser, John G. Palfrey: Case Study: Mashups Interoperability and eInnovation. Berkman Center Research Publication No. 2007-10.
- [8] Spoerri, A.: Visual Mashup of Text and Media Search Results. In: *11th International Conference Information Visualization*, pp. 216-222, 2007.
- [9] Roy T. Fielding, *Architectural Styles and the Design of Network-based Software Architecture*, 2000.
- [10] T. Kikukawa, S. Kawafuchi, "Development of an automatic summary editing system for the audio-visual resources", *Transactions on Electronics and Information J75-A*, pp. 204-212, 1992.
- [11] H. Zhang, A. Kankanhalli and W. Smoliar, "Automatic partitioning of full-motion video", *Multimedia Systems*, vol.1, no.1, pp. 10-28, 1993.
- [12] A. Nagasaka, Y. Tanaka, "Automatic video indexing and full-video search for object appearances", in *Visual Database Systems II* (E. Knuth and L.M. Wegner, eds.), pp. 113-127, Elsevier, 1995.
- [13] R. Kasturi, R. Jain, "Dynamic vision, in *Computer Vision: Principles*", R. Kasturi and R. Jain, (eds.), pp. 469-480, IEEE Computer Society Press, Washington DC, 1991.
- [14] B. Shahraray, "Scene change detection and content-based sampling of video sequences", in *Proc. IS&T/SPIE 2419*, pp. 2-13, 1995.
- [15] W. Xiong, J. C.-M. Lee, "Efficient scene change detection and camera motion annotation for video classification", *Computer Vision and Image Understanding*, vol. 71, no. 2, pp. 166-181, 1992.
- [16] W. Xiong, J. C.-M. Lee, M.C. Ip, "Net comparison: a fast and effective method for classifying image sequences", in: *Proc. SPIE Conf. Storage and Retrieval for Image and Video Databases III 2420*, San Jose, CA, pp. 318-328, 1995.

public

- [17] G. Pass, R. Zabih, “Comparing images using joint histograms”, *Multimedia Systems*, Springer-Verlag, vol.7, no.3, pp.234-240, May 1999.
- [18] U. Gargi, R. Kasturi, S. Antani, “Performance characterization and comparison of video indexing algorithms”, in: *Proc. Conf. Computer Vision and Pattern Recognition (CVPR)*, 1998.
- [19] H. Zhang, “Content-based Video Browsing and Retrieval”, from “*Handbook of Multimedia Computing*” editor-in-chief Furht B., CRC Press, Boca Raton, Florida, USA, 1999.
- [20] J. Boreczky, L.A. Rowe, “Comparison of video shot boundary detection techniques”, in: *Proc. IS&T/SPIE Intern. Symposium Electronic Imaging*, San Jose, 1996.
- [21] D. Swanberg, C.-F. Shu, and R. Jain, “Knowledge guided parsing in video databases,” in *Proc. IS&T/SPIE Symp. Electronic Imaging: Science and Technology*, San Jose, CA, Feb. 1993.
- [22] C. Lee, D. Ip, “A robust approach for camera break detection in color video sequences”, in: *Proc. IAPR Workshop Machine Vision Appl.*, Kawasaki, Japan, pp. 502-505, 1994.
- [23] T. Pappas, “An adaptive clustering algorithm for image segmentation”, *IEEE Trans. on Signal Processing* 40, pp. 901-914, 1992.
- [24] A. Ferman, A. Tekalp, “Efficient filtering and clustering for temporal video segmentation and visual summarization”, *Journal of Visual Communication and Image Representation*, vol. 9, no. 4, pp. 3368-351, 1998.
- [25] R. Zabih, J. Miler, K. Mai, “A feature-based algorithm for detecting and classifying production effects”, *Multimedia Systems* 7, pp. 119-128, 1999.
- [26] A. Hampapur, R. Jain, T. E. Weymouth, “Production model based digital video segmentation”, *Multimedia Tools and Applications*, vol. 1, no. 1, pp. 9-46, 1995.
- [27] P. Aigrain, P. Joly, “The automatic real-time analysis of film editing and transition effects and its applications”, *Computers and Graphics*, vol. 18, no. 1, pp. 93-103, 1994.
- [28] H. Yu, G. Bozdagi, S. Harrington, “Feature-based hierarchical video segmentation”, in: *Proc. Int. Conf. on Image Processing (ICIP'97)*, Santa Barbara, pp. 498-501, 1997.
- [29] J. Boreczky, L.D. Wilcox, “A hidden Markov model framework for video segmentation using audio and image features”, in: *Proc. Int. Conf. Acoustics, Speech, and Signal Proc.*, 6, Seattle, pp. 3741-3744, 1998.
- [30] J. Shi, and J. Malik, “Normalized Cuts and Image Segmentation”, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888-905, 2000.
- [31] B.S. Manjunath, P. Salembier, T. Sikora, “Introduction to MPEG-7, *Multimedia Content Description Interface*”, John Wiley & Sons, 2003.
- [32] M. Meila and J. Shi, “A Random Walks view on Spectral Segmentation”, *Artificial Intelligence and Statistic Workshop AISTATS*, 2001
- [33] M. Meila, M. S. Shortreed, and L. Xu, “Regularized Spectral Learning”. *Proceedings of the Artificial Intelligence and Statistics AISTATS*, 2005.
- [34] Y. Yusoff, W. J. Christmas and J. Kittler, “Video shot cut detection using adaptive threshold”, *BMVC* 2000.