TABLE OF CONTENTS

Page

INTRODUCTION 1

CHAPTER 1 BACKGROUND 9
1.1 Multimedia Copy Detection Concepts 9
1.1.1 Multimedia Copy Detection Terminology 9
1.1.2 Structure of a Multimedia Fingerprinting System 10
1.1.3 Properties of a Multimedia Fingerprint 11
1.1.4 System Requirements 12
1.1.5 Applications of Multimedia Copy Detection 13
1.2 Datasets and Evaluation Metrics 15
1.2.1 Datasets 15
1.2.2 Evaluation metrics 19
1.3 Related Work 21
1.3.1 Audio Fingerprinting 22
1.3.2 Video Fingerprinting 26
1.3.3 Audio+Video Systems 29
1.3.4 Accelerating Fingerprints Search 31
1.4 Summary 34

CHAPTER 2 AUDIO FINGERPRINTING 37
2.1 System Overview 37
2.2 Spectrogram Generation 38
2.3 Fingerprint Generation 38
2.3.1 Global Mean Fingerprint 39
2.3.2 Local Mean Fingerprint 41
2.3.3 Salient-Regions Fingerprint 43
2.4 Query Fingerprint Generation 45
2.5 Retrieval 46
2.5.1 Similarity Search 46
2.5.1.1 Similarity Measure for Global Mean and Local Mean Fingerprints 47
2.5.1.2 Similarity Measure for Salient-Regions Fingerprint 47
2.5.2 Matching 48
2.6 Results and Analysis 50
2.6.1 Results for Global Mean Fingerprint 51
2.6.2 Results for Local Mean Fingerprint 57
2.6.3 Combined Results from Global Mean and Local Mean Fingerprints 60
2.6.4 Results for Salient-Regions Fingerprint 61
2.6.5 Comparative Audio Copy Detection Systems 64
2.7 Summary 68

CHAPTER 3 VIDEO FINGERPRINTING 71
3.1 System Overview 71
3.1.1 Letterbox Detection 72
3.1.2 PiP Detection 73
3.1.3 Video Fingerprint Extraction 75
3.1.4 Matching Algorithm 78
3.2 Results and Analysis 79
3.2.1 Video Only Results 79
3.2.2 Audio+Video Results 84
3.3 Summary 86

CHAPTER 4 ACCELERATING THE AUDIO FINGERPRINT SEARCH USING A GPU AND A CLUSTERING-BASED TECHNIQUE 89
4.1 GPU Implementation of the Similarity Search 89
4.1.1 GPU Architecture 90
4.1.2 Similarity Search on GPU 92
4.1.3 Similarity Algorithms 94
4.2 Clustering-Based Technique 96
4.2.1 Training 97
4.2.2 Lookup Table Construction 99
4.2.3 Matching Algorithm 99
4.2.4 Two-step Search 100
4.3 Results and Analysis 101
4.3.1 Run Times of GPU Implementations 102
4.3.2 Clustering-based Technique Performance 109
4.3.3 Validation Results on TRECVID 2009 115
4.3.4 Scalability 115
4.3.5 Shazam versus CSR-44 116
4.4 Summary 118

CONCLUSION 121

LIST OF REFERENCES 127

LIST OF TABLES

Page

Table 1.1 Description of audio and video transformations 16

Table 2.1 Percentage of 1's in the binary images averaged over all query/reference frames for Global Mean fingerprints 52

Table 2.2 Performance of Global Mean with varying dimensions for different transforms 53

Table 2.3 Min NDCR generated with Global Mean fingerprint using different SCF values 55

Table 2.4 Min NDCR per transformation obtained with Global Mean feature with/without using binary images and with only one fingerprint version instead of four versions 55

Table 2.5 Min NDCR, number of missed queries and Mean F1 for Global Mean fingerprint on TRECVID 2009 57

Table 2.6 Min NDCR for Local Mean fingerprint for different thresholds and different SCF values when tested on TRECVID 2010 dataset 58

Table 2.7 Min NDCR for combined features on TRECVID 2010 dataset 60

Table 2.8 Min NDCR for Salient-Regions fingerprint averaged over all transformations with varying SCF values 62

Table 2.9 Min NDCR for Salient-Regions (SR) fingerprint with varying dimensions compared to Global Mean (GM) and Local Mean (LM) fingerprints 63

Table 3.1 Min NDCR by transformation achieved by different visual features on TRECVID 2009 and 2010 datasets 80

Table 3.2 Number of missed queries for V-intensity and V-motion on TRECVID 2009 dataset 82

Table 3.3 PIP detection performance 82

Table 3.4 Min NDCR per transformation for audio+video with SR-44 system on TRECVID 2009 dataset 85

Table 3.5 Min NDCR per transformation for audio+video with SR-44 system on TRECVID 2010 dataset 85

Table 4.1 CPU and GPU processing run time in seconds for hashing-based and sorting-based algorithms using 10 queries 102

Table 4.2 Processing run time using 10 queries of hashing-based and sorting-based algorithms with and without using shared memory 103

Table 4.3 Hashing-based and sorting-based run times in seconds with different configurations using 10 queries 104

Table 4.4 Proposed system performance using varying dimensions when tested with all the queries on TRECVID 2010.....109

Table 4.5 Number of missed queries by transformation for SR-44 and CSR-44 110

Table 4.6 Min NDCR by transformation for SR-44 and CSR-44.....110

Table 4.7 Min NDCR of NN-based, Shazam, Salient Regions (SR) and the best results achieved using the clustering technique (CSR) with 12 and 44 dimensions113

Table 4.8 Min NDCR for SR-44 and CSR-44 compared to the WASF and Coherency method on TRECVID 2009 dataset 115

Table 4.9 Average run time (secs) on GTX 580 and GTX TiTan X for SR-44 and CSR-44 top 300 segments on the doubled reference dataset.....116

Table 4.10 Mean F1 for the proposed system (CSR-12 with Top 200 segments), NN-based and Shazam systems 118

LIST OF FIGURES

Page

Figure 1.1 Overall structure of a multimedia fingerprinting system 11

Figure 1.2 Examples of TRECVID video transformations....17

Figure 1.3 Query creation framework....18

Figure 2.1 Proposed audio fingerprinting system overview 38

Figure 2.2 Global Mean fingerprint extraction 39

Figure 2.3 Four different versions of quantized spectrogram matrix generated for four different audio frames (1-sec window frame)....40

Figure 2.4 Local Mean feature extraction....41

Figure 2.5 Global Mean versus Local Mean Fingerprints 42

Figure 2.6 Fingerprint representation of Global Mean and Local Mean fingerprints 43

Figure 2.7 Feature extraction with 16 tiles 45

Figure 2.8 Matching frames based on nearest neighbor search 48

Figure 2.9 Improved matching frame algorithm....50

Figure 2.10 Example of the repeated segment problem 59

Figure 2.11 Comparison of the best results for each feature separately and for their combination....61

Figure 2.12 Min NDCR for Salient-Regions when extracting features from binary images or spectrogram matrix, for (a) 24 dimensions and (b) 44 dimensions 63

Figure 2.13 Salient-Regions, NN-based and Shazam evaluation on TRECVID 2010 dataset: (a) min NDCR and (b) Mean F1 64

Figure 2.14 Average run time (in secs/query) versus min NDCR (averaged over all transformations) for different systems on TRECVID 2010 dataset....65

Figure 2.15 Number of missed queries per transformation for different systems on TRECVID 2009 dataset 66

Figure 3.1 Proposed video copy detection system overview....72

Figure 3.2 PiP detection steps 74

Figure 3.3 Example of PiP detection 75

Figure 3.4 Illustration of V-intensity fingerprint generation scheme 77

Figure 3.5 Illustration of V-motion fingerprint generation scheme 78

Figure 3.6 Example of PiP detection problem 83

Figure 3.7 Min NDCR of the proposed system for audio+video transformations compared to Perseus system on TRECVID 2010 dataset 86

Figure 4.1 CUDA Hierarchy of threads, blocks, and grids, with corresponding per-thread private, per-block shared, and per-application global memory spaces 91

Figure 4.2 Processing flow on CUDA 92

Figure 4.3 Processing steps on GPU 96

Figure 4.4 Illustration of the main components of the clustering-based technique 97

Figure 4.5 Illustration of the two-step search. The Top N results generated from step 1 are rescored in step 2 for accurate results 101

Figure 4.6 Impact of the number of dimensions of the reference frame on (a) run time, (b) number of used registers and (c) amount of used local memory 108

Figure 4.7 Performance of the clustering-based technique: (a) impact of the number of clusters on min NDCR; (b), (c) and (d) evolution of run time, min NDCR and missed queries, respectively, for CSR-44 with Top N files and segments ((e), (f) and (g) for CSR-12) 111

LIST OF ABREVIATIONS

BCS	Bounded Coordinate System
CPU	Central Processing Unit
CSR	Clustering-Based technique
CUDA	Compute Unified Device Architecture
CBCD	Content-Based Copy Detection
FA	False Alarm
FN	False Negative
FP	False Positive
GM	Global Mean
GPU	Graphics Processing Unit
HAS	Human Auditory System
IDC	International Data Corporation
LM	Local Mean
LSH	Locality Sensitive Hashing
LUT	LookUp Table
MFCC	Mel-Frequency Cepstral Coefficients
MPI	Message Passing Interface
NIST	National Institute of Standards and Technology
NOFA	No False Alarm
NDCR	Normalized Detection Cost Rate
PIP	Picture in Picture

RIAA	Recording Industry Association of America
SR	Salient-Regions
SIFT	Scale Invariant Feature Transform
SM	Shared Memory
SIMT	Single Instruction Multiple Threads
SMP	Streaming MultiProcessor
SCF	Successive Closest Frames
TIRI	Temporally Informative Representative Images
TSM	Time Scale Modification
TN	True Negative
TP	True Positive
VQ	Vector Quantization
WASF	Weighted Audio Spectrum Flatness

INTRODUCTION

Advancements in information technology since the beginning of the digital revolution in the late 1950s have incorporated multimedia in all aspect of our life. Specifically, the innovations in telecommunications, signal compression, data transmission and the increase in computing and storage capacities have made the production, distribution and reproduction of multimedia content not only an easy task, but also a widespread phenomenon ubiquitous in our social and professional lives. The increasing popularity of video sharing web sites, such as YouTube, Dailymotion and Metacafe, shows the extent of this reality. For instance, YouTube claims that, in March 2015, 300 hours of videos were uploaded to its platform every single minute (Youtube, 2015). Though this huge data traffic reflects user appreciation, it also implies serious copyright issues. In fact, a large number of the uploaded content may be illegal copies of digital material protected by copyright law. To deal with this, YouTube has invested millions of dollars into their Content ID copyright management system, and has paid out over $1 billion to partners who have chosen to monetize their claims using Content ID (Youtube, 2015). Besides, file-sharing networks (peer-to-peer) have cost the music industry billions in economic losses. In fact, according to the Recording Industry Association of America (RIAA), 30 billion songs were illegally downloaded on file-sharing networks between 2004 and 2009 (Riaa, 2015). These facts have made copyright infringement one of the biggest issues that hosting web sites, file-sharing networks and similar services have to deal with to avoid lawsuits by the copyright holders.

To cope with this problem, Content-Based Copy Detection (CBCD) has been recently proposed as an alternative solution to the watermarking technique. The watermarking technique inserts invisible information into the original document to allow subsequent detection of this document. The major limitation of watermarking is its spread: it is impossible to detect documents that are not watermarked. The idea behind CBCD is that the content itself contains enough unique information that it can be used as a *watermark* to detect copies. Hence, a multimedia document can be detected solely based on its content without altering the original signal by adding a watermark.

Besides being a successful solution to control illegal distribution of copyrighted content, content-based copy detection is suitable for a wide variety of applications such as audio and video indexing, broadcast monitoring, music identification, multimedia library organization, advertisement detection, etc.

Problem statement

A multimedia copy detection system provides the ability to automatically determine if a multimedia query (audio/video file or part of it) is a copy of a multimedia document derived from a large, already known, multimedia references collection. Traditional solution is to insert hidden information, called watermark, to the original signal (Hartung and Kutter, 1999). The authenticity of a digital document is then verified based on the presence of the watermark. This approach has been used for many years to prevent copyright infringements of electronic text documents, digital images, audio and video documents. The main advantages of watermarking approach are the possibility to encode additional information into the watermark (e.g. identity of the owner), and its efficiency in terms of processing run time. However, watermarking techniques are unable to detect content that are not watermarked before their propagation. Furthermore, watermarking techniques are vulnerable to many malicious attacks that can prevent detection of the watermark (Le, Nguyen and Le, 2010; Tanha et al., 2012; Voloshynovskiy et al., 2001). In addition, the extra information inserted into the signal usually affects the quality of the watermarked content.

To cope with these limitations, content-based copy detection has been recently introduced (Cano et al., 2002; Cano et al., 2005) (Haitsma and Kalker, 2002) as an alternative or a complementary technique to the watermarking approaches. Instead of inserting additional information into the content, CBCD uses the content itself as a watermark. It extracts relevant features (often called signatures or fingerprints) from a candidate copy and then compares them against fingerprints of the original content.

The task would be straightforward if the two signals were exactly the same. However, audio and video signals are subjected to various kinds of transformations that make robust fingerprint extraction challenging. Audio transformations include audio compression, signal degradation and addition of irrelevant speech or noise. A video copy can be modified by more diversified transformations such as insertion of pattern, picture in picture, decrease of quality, re-encoding or even a combination of different transformations.

The robustness of fingerprints against audio transformations is an important element of any audio identification system. In fact, an ideal fingerprinting system should be able to detect a copy of an original multimedia file regardless of the nature and the complexity of the transformations (Cano et al., 2002; Cano et al., 2005). Another essential requirement that should be taken into consideration is the search efficiency. The speed of the fingerprints search algorithm is becoming increasingly important due to the large volume of data. Thus, the search of an audio/video against a large set of fingerprints should be at least real time.

Objectives of the thesis

The overall objective of this thesis is to design a robust and efficient content-based multimedia copy detection system. This system is composed of two parts: audio copy detection and video copy detection. Each subsystem works independently allowing the detection of a multimedia document based on either audio or video (visual-based) fingerprints. In addition, we propose to implement a fusion strategy that combines audio and video results generated separately in order to detect video copies transformed by audio and video transformations.

Regardless of the types of fingerprints (audio or video), the proposed system should be able to automatically detect a transformed copy of multimedia object contained in a large dataset of reference multimedia objects. Besides, the system should be able to correctly detect transformed copies of any duration, from a few seconds to several minutes. When a copy is detected, the system should indicate the exact location of the query within the reference.

Finally, the processing run time needed to perform the search of a given query should be at least real time. The latter constraint is imposed by the quantity of data to process and the need to quickly detect the spread of a fraudulent document.

To sum up, this thesis attempts to achieve two specific objectives that ensure two principal requirements of a multimedia copy detection system: robustness and efficiency. The first objective is to define a fingerprint extraction scheme that generates robust audio and video fingerprints to different forms of signal transformations. The second objective concerns the efficiency of the system, and involves using an effective strategy that ensures a very fast retrieval of fingerprints from a large reference dataset.

Contributions

The majority of state-of-the-art audio fingerprinting systems are based on binary fingerprint representation. The *energy difference fingerprint* (Haitsma and Kalker, 2002) is among the first methods published in this field where audio frames are transformed into binary codes. Many other works based on the energy difference fingerprints or similar approaches to generate binary fingerprints have been proposed (Haitsma and Kalker, 2002; Lebosse, Brun and Pailles, 2007; Lezi et al., 2012; Saracoglu et al., 2009). The popularity of binary fingerprints mainly stems from feature extraction simplicity and fingerprint retrieval efficiency. In fact, finding fingerprints based on a lookup table is a trivial task and results in a very fast search. However, different binary codes may be generated from a given reference frame and its transformed copy, which results in a significant loss in detection performance.

On the other hand, methods that do not use binary fingerprints are, in general, more robust to audio degradations. An example of such a system is the NN-based system (Gupta, Boulianne and Cardinal, 2012), which is based on a nearest neighbor mapping between query frames and reference frames. These fingerprint retrieval algorithms need to perform frame-by-frame computations, making the search very time-consuming. In fact, it is common in these fingerprinting systems to trade off detection performance for speed, or vice-versa.

As mentioned before, in our work we propose a fingerprinting system that combines robustness and efficiency. We will handle the robustness aspect of our system by implementing an audio fingerprint generation scheme in an unconventional way. We propose to achieve the robustness by generating different versions of the spectrogram matrix of the audio signal. Then, we transform each version of this spectrogram matrix into a 2-D binary image. Multiple versions of these 2-D images suppress noise to a varying degree. This varying degree of noise suppression improves the likelihood of one of the images matching a reference image. The novelty of this approach lies in the conversion of the spectrogram into a set of binary images and the derivation of multiple fingerprints from these images. Based on this strategy, we will describe two fingerprint extraction methods. In addition, we will introduce a third audio fingerprint extraction method that selects several salient regions from the binary images. In this method, each fingerprint encodes the positions of the selected salient regions. The advantage of this approach is that it allows encodes the most relevant information of the signal without encoding the energy of the signal. Therefore, it makes the fingerprints robust to many transformations that change the energy of the signal.

Second, we will address the video copy detection problem by proposing a visual feature extraction algorithm which uses a similar strategy to that of the audio fingerprints by encoding the positions of salient regions. However, instead of selecting the salient regions from binary images, video-based salient regions are selected from each grayscale video image. We will describe two methods based on this approach that generate two different video fingerprints. To address the problem of audio+video copy detection where the queries are transformed by a combination of audio and video transformations, we propose a merging method that combines the results obtained separately from the audio and the video parts.

In regards to the efficiency of the system, we propose two different techniques to speedup the search of fingerprints. The first technique is based on a hardware implementation using a Graphics Processing Unit (GPU) to parallelize the computation of the similarity between fingerprints. We will present GPU implementations of two similarity search algorithms that perform fingerprint retrieval efficiently in the context of a large fingerprints dataset. The

second technique is based on a software solution. We propose a two-step search algorithm based on a clustering technique and a lookup table that reduces the number of comparisons between the query and the reference fingerprints.

Outline of the thesis

This thesis is organized into four chapters. Chapter 1 introduces the content-based multimedia copy detection field by presenting different related aspects. We describe the datasets and the metrics used to evaluate all systems we proposed. In the last part of this chapter we give an overview of the relevant state-of-the-art methods related to audio and video content-based copy detection.

Chapter 2 addresses the problem of content-based audio copy detection. We describe our novel audio fingerprint extraction approaches and the fingerprint retrieval algorithm. We present and discuss the results obtained from our audio fingerprinting system while comparing it to a number of state-of-the-art audio copy detection systems. Different parts of this chapter were published in three conferences: (Ouali, Dumouchel and Gupta, 2014a), (Ouali, Dumouchel and Gupta, 2014b), and (Ouali, Dumouchel and Gupta, 2015b). In addition, a journal paper was published in the Multimedia Tools and Applications (Ouali, Dumouchel and Gupta, 2015e).

Chapter 3 addresses the problem of content-based video copy detection. We present our video copy detection system and all its components including two new video fingerprint extraction methods. We evaluate the performance of the proposed system to detect video only copies and audio+video copies by combining audio and visual features. We also compare the performance of this system to state-of-the-art video fingerprinting methods. A part of this chapter was published in (Ouali, Dumouchel and Gupta, 2015a).

In Chapter 4 we improve the performance of our audio fingerprinting system in terms of processing run time. We present a detailed parallel design of a fingerprint similarity search

algorithm suitable for a large dataset of fingerprints. Besides, we introduce a new two-step search approach based on clustering and table lookup. We evaluate the performance of the proposed GPU system in terms of efficiency. In addition, we evaluate the performance of our system when the clustering-based technique is used and we discuss the tradeoff between the speed of the search and the copy detection performance. We also compare our system to other audio copy detection systems. This chapter was submitted to IEEE Transactions on Audio, Speech and Language Processing (Ouali, Dumouchel and Gupta, 2015c), and a part of it was published in (Ouali, Dumouchel and Gupta, 2015d).

In the general conclusion, we summarize the work accomplished this thesis, and we provide recommendations and perspectives for future work.

CHAPTER 1

BACKGROUND

The present chapter is an introduction to the multimedia content-based copy detection field. The aim of the first section is to give a global vision on the multimedia fingerprinting technology by covering different aspects including the multimedia copy detection problem and the related terminology. We also describe the design and requirements of a content-based multimedia copy detection system as well as some applications that can benefit from the use of the fingerprinting technology. The second section describes the platform (datasets and evaluation metrics) that we used to evaluate our work outlined in the next three chapters. The third section reviews previous approaches in the area of audio and video content-based copy detection.

1.1 Multimedia Copy Detection Concepts

Multimedia copy detection is an active research area that gains paramount importance as audio and video content has become increasingly an integral part of all aspects of our lives. However, this field is relatively new, and several principles should be described in order to have a complete vision on all sides of this emerging technology. Thus, the aim of this section is to clarify some of the major concepts related to multimedia copy detection, while presenting its utility through several examples of application scenarios.

1.1.1 Multimedia Copy Detection Terminology

The task of multimedia copy detection is to automatically determine whether a given multimedia object *A* (image, audio or video) is similar to another object *B* that is part of a large collection of multimedia objects. In the literature, object *A* is often called a *copy* or *duplicate* of an *original* content (i.e. object *B*). A major difficulty of the copy detection problem is that the copied segment is not an exact reproduction of the original content, but

rather a transformed copy. This is why the term *near-duplicate* is also used to refer to the copied segment.

Another term commonly used to describe the task of detecting a multimedia copy is *Content-Based Copy Detection* (CBCD). This term is used to describe a specific copy detection approach that uses the content itself of the multimedia document to detect a copy, in contrast to the *watermarking* approach that inserts a watermark into the original digital document to allow its subsequent detection.

The CBCD approach is also widely known as *multimedia fingerprinting* since specific features called *fingerprints* are extracted from the multimedia content and used thereafter to establish the similarity of two digital documents. Although CBCD and multimedia fingerprinting are the most common terms used to refer to this approach, the research literature includes other names such as *robust hashing*, *perceptual hashing*, *robust matching,* and *passive watermarking*.

1.1.2 Structure of a Multimedia Fingerprinting System

A common design of a multimedia fingerprinting system includes two principal components: (1) a method to extract fingerprints from an audio/video signal that describes specific properties of the signal; (2) a method to search fingerprints of an unknown audio/video in a large dataset of fingerprints. An illustration of such design is shown in Figure 1.1, where two stages describing the system functionality can be identified.

During the first stage, the system extracts fingerprints from each audio/video reference file and stores them into a reference fingerprints database. In the second stage, the system extracts fingerprints from the audio/video query using the same extraction algorithm used to extract reference fingerprints, and scans the reference fingerprints database to find potentially similar fingerprints. Unlike the first stage that could be processed a posteriori (off-line), the second stage is performed on-line affecting subsequently the system efficiency.

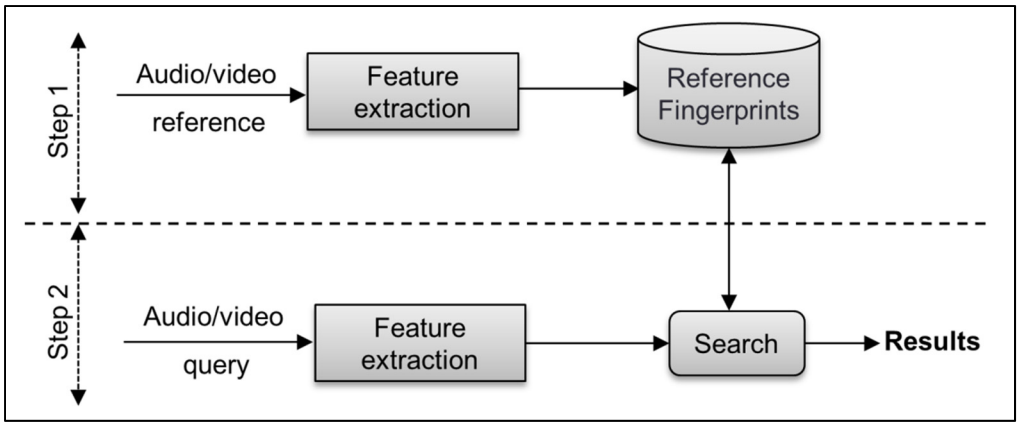

Figure 1.1 Overall structure of a multimedia fingerprinting system

It is worth noting that the architecture described above is just an overall structure of a multimedia fingerprinting system. A more realistic architecture includes additional components depending on the kind of the multimedia objects to be detected (audio or video), the nature and complexity of the transformations applied to the query, the feature extraction approach, the fingerprint matching algorithm, etc.

1.1.3 Properties of a Multimedia Fingerprint

A multimedia fingerprint is a unique identifier representing a "signature" extracted from the content of an audio/video signal. In general, it represents the basic element of a multimedia fingerprinting system, on which the rest of the system is built.

The robustness and efficiency of the system are considerably influenced by the design of the fingerprints. Thus, the feature extraction approach should be designed in a way to guarantee the generation of fingerprints that hold specific characteristics. Regardless of the feature extraction approach, a fingerprint should ideally have the following properties:

- **Robustness**: The robustness of a fingerprint represents its resistance to the presence of signal degradation. Thus, a fingerprint generated from the original audio/video

content should be similar to the fingerprint generated from a distorted copy of the same content.

- **Uniqueness**: The uniqueness of a fingerprint reflects its discrimination capability over a large number of fingerprints. This property ensures that fingerprints generated from two different signals are distinct. The robustness and uniqueness of a fingerprint are two conflicting properties, and usually a trade-off is made between them. In fact, increasing the invariability of the fingerprint to signal degradation decreases its sensitivity to signal change (i.e. the fingerprint becomes less discriminative). On the other hand, increasing the discrimination ability of a fingerprint decreases its ability to survive in presence of noise.
- **Compactness:** The size of a fingerprint can have significant impact on the memory/storage requirements and the system efficiency. Thus, the fingerprint representation should be small so as to decrease the complexity of the system and reduce the memory space required to store a large number of fingerprints. However, fingerprint representation should at the same time contain the most relevant signal information that maintains the robustness and the discrimination power of the fingerprint.
- **Easy to compute**: The fingerprint extraction algorithm should have low computational complexity. This fingerprint property is very important, especially for on-line applications that require real-time detection.

1.1.4 System Requirements

Whether a multimedia fingerprinting system is designed to detect audio or video content, it should meet several requirements to be considered suitable for real world applications. A large number of requirements have been identified in several papers (Cano et al., 2005; Haitsma and Kalker, 2002; Lu, 2009b)**.** The main requirements of a multimedia copy detection system include:

- **Robustness**: An ideal multimedia copy detection system should be able to detect a transformed copy regardless of the nature and complexity of the audio and/or video transformations.
- **Accuracy**: The level of accuracy describes the ability of the system to properly identify a copy and its capacity to reject a non-copy. The accuracy of a multimedia copy detection system can also be referred to its ability to locate the start and the end of the copied segment in a reference audio/video file (i.e. time localization accuracy).
- **Reliability**: This requirement reflects the ability of the system to correctly reject a query (referred to as true negative) that does not exist in the reference database. Besides, identifying the wrong item from the reference database (referred to as false positive) instead of the correct item decreases the reliability of the system.
- **Granularity**: The number of seconds needed to identify a copy defines the granularity of the system. The granularity of a multimedia system is an important requirement for many applications since it often increases the complexity of the system.
- **Efficiency**: The search speed of a multimedia copy is a key requirement for most of the multimedia copy detection systems, especially for those that require a real time response or have limited computing resources.
- **Scalability**: The multimedia copy detection system should be scalable to a large multimedia database, while maintaining good performance in terms of detection and efficiency.

1.1.5 Applications of Multimedia Copy Detection

Multimedia copy detection field has seen a growing scientific interest in the last decade resulting in significant performance improvements. Research in this area has increased the industrial interest to exploit this technology to create practical applications. In fact, multimedia fingerprinting technology has allowed the development of several real-world applications, where some of them have been successfully commercialized. We enumerate in the following three application scenarios where this technology can be applied.

- **Copyright protection**: Automatically detecting illegal copies of protected digital content is among the applications where the multimedia fingerprinting technology offers an excellent solution. Nowadays, most of the content providers adopt the multimedia fingerprinting technology to detect and filter illegal copies. Examples of these content providers include YouTube (Youtube, 2015), Vimeo (Vimeo, 2015), Yahoo! (Yahoo, 2015) and Dailymotion (Dailymotion, 2015). In addition, a large number of companies provide their services to automatically monitor a large number of media sharing platforms in order to detect illegal copies of protected content (e.g. music, films, TV shows, etc.). Audible Magic (Audiblemagic, 2015) is an example of such companies; their multimedia copy detection product is used by a large number of companies like Facebook (Facebook, 2015), Disney (Disney, 2015) and SoundCloud (Soundcloud, 2015)**.**
- **Broadcast monitoring**: Advertisers are interested to monitor radio, TV and web broadcasts to track their advertisements and verify if they are being broadcasted as agreed. Using the multimedia fingerprinting technology also allows the companies to follow advertising campaigns of their competitors for business intelligence purposes. Vobile (Vobile, 2015) is among the companies that provide this kind of services. Their products are based on the use of the multimedia fingerprinting technology to automatically identify audiovisual content.
- **Music identification**: The popularity of smartphones together with the maturity of multimedia fingerprinting technology have allowed this kind of application to be a huge success with music lovers. This kind of application recognizes a song on real time using an intelligent mobile phone that records a small segment of the music being played. The application provides to the users all the information related to the recognized song, and enables the user to directly buy the song. Shazam (Shazam) is the best-known company that offers this service with more than 500 million users.

The application scenarios listed above have gained significant benefit from this technology. Nevertheless, the multimedia fingerprinting technology can be applied in other applications such as multimedia library organization or *query by video clip* search.

1.2 Datasets and Evaluation Metrics

A large number of the proposed approaches in the area of content-based multimedia copy detection have been evaluated using uncommon evaluation framework, where different private corpus of various kinds have been used. In addition, a variety of evaluation metrics have been used to evaluate the effectiveness and the detection performances. These ways of doing do not allow an objective comparison of fingerprinting systems. It could also leads to a misinterpretation of the system performances. For this reason, all our experiments are conducted on the well-known and standard TRECVID copy detection datasets (Awad, Over and Kraaij, 2014).

TRECVID datasets are provided by the National Institute of Standards and Technology (NIST) as part of the content-based copy detection track organized in context of the international conference on benchmarking for content-based video indexing and retrieval. In addition to the datasets, NIST provides an evaluation platform where different metrics are applied with varying parameters to evaluate the performance of a copy detection system in context of two application scenarios. We describe in the following the TRECVID datasets as well as the evaluation metrics used in our work.

1.2.1 Datasets

Our evaluations are conducted using two datasets: TRECVID 2009 and TRECVID 2010 copy detection datasets. We briefly present these two datasets; a detailed description of these datasets can be found in (Awad, Over and Kraaij, 2014).

TRECVID 2009 dataset contains 385 hours of reference videos composed of news magazine, science news, documentaries, educational and historical video obtained from the *Netherlands Institute for Sound and Vision*. The non-reference videos are raw material collected from several BBC programs.

TRECVID 2010 consists of a reference collection of more than 11,000 videos for a total of 400 hours of videos. In this dataset, both reference and non-reference collections are composed of a vast variety of videos downloaded from the Internet.

For each dataset, there are 201 different original queries, each query altered with seven audio transformations (for a total of 1407 audio queries) and eight video transformations (for a total of 1608 video queries). Descriptions of audio and video[1] transformations are shown in Table 1.1.

Table 1.1 Description of audio and video transformations

Type	Label	Description
Audio transformation	T1	Nothing
	T2	mp3 compression
	T3	mp3 compression and multiband companding
	T4	bandwidth limit and single band companding
	T5	mix with speech
	T6	mix with speech, then multiband compress
	T7	bandpass filter, mix with speech, compress
Video transformation	V1	Simulated camcording
	V2	Picture in picture type 1: original video in front of background video
	V3	Insertions of pattern
	V4	Strong re-encoding
	V5	Change of gamma
	V6	Decrease in quality: introducing three randomly selected combinations of Blur, Gamma, Frame dropping, Contrast, Compression, Ratio, White noise

[1] Video transformations V1, V7 and V9 are not used in TRECVID 2009 evaluation campaign because they were extreme. For the same reason, NIST has also dropped V7 and V9 (these two transformations are not described in Table 1.1) from TRECVID 2010 evaluation campaign.

Table 1.1 Description of audio and video transformations

Type	Label	Description
Video transformation	V8	Post production: introducing three randomly selected combinations of Crop, Shift, Contrast, Text insertion, Vertical mirroring, Insertion of pattern, Picture in picture
	V10	Combination of three randomly selected transformations chosen from V1-V8

Figure 1.2 shows an example of a video frame (*original*) transformed with eight video transformations described in Table 1.1.

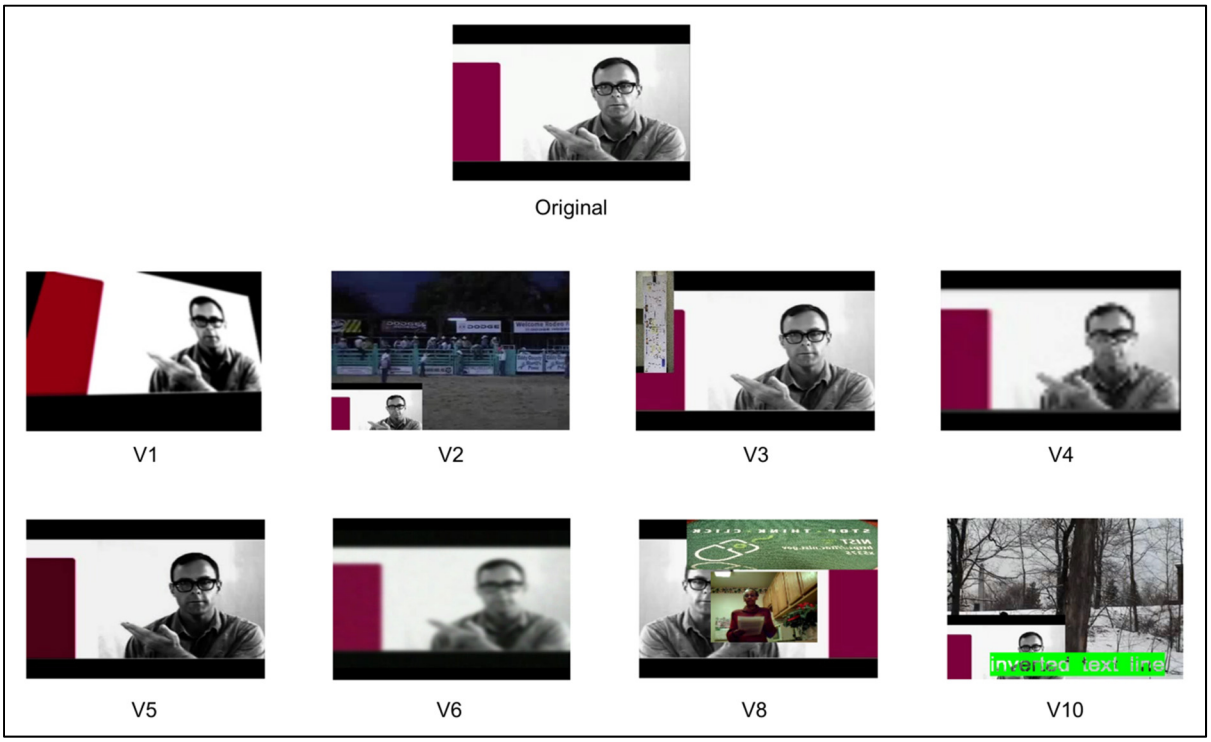

Figure 1.2 Examples of TRECVID video transformations

V1 transformation is not applied in TRECVID 2009, which results in 1407 video transformations (compared to 1608 video queries in TRECVID 2010 dataset). An illustration of the query creation framework is shown in Figure 1.3. More details can be found in (NIST, 2015).

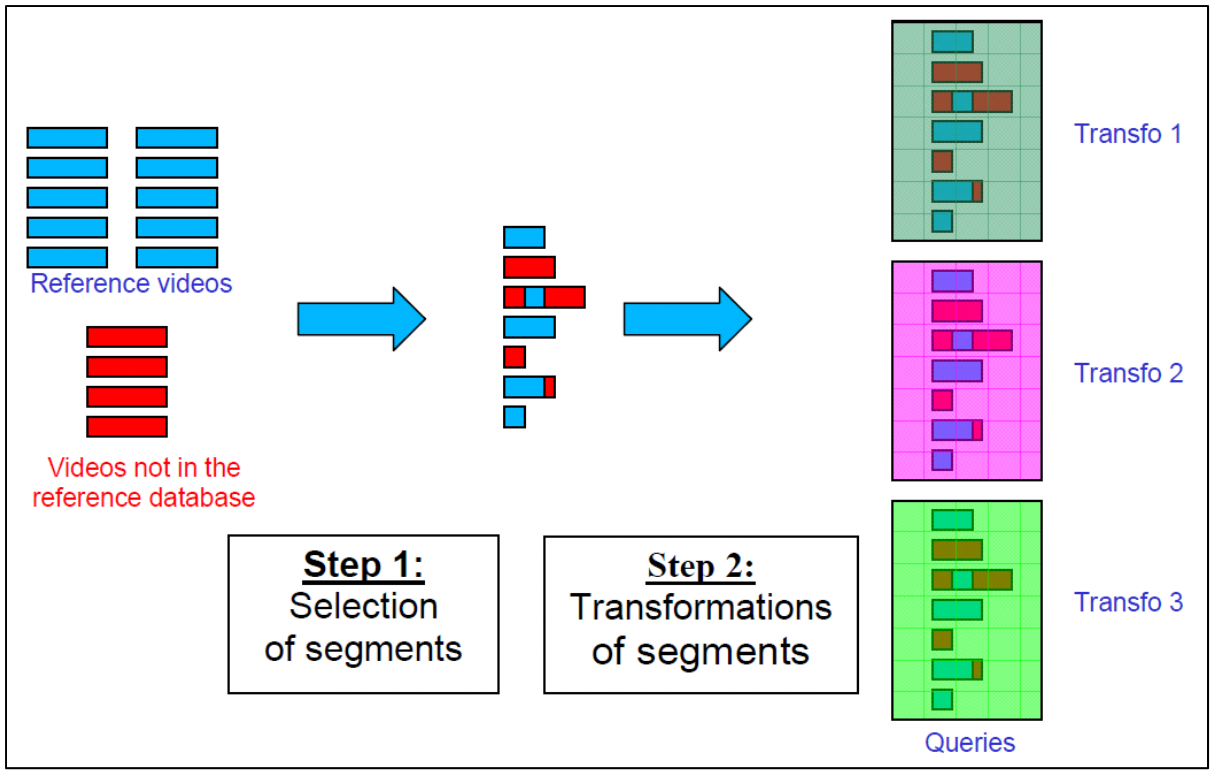

Figure 1.3 Query creation framework
Taken from (NIST, 2015)

The length of queries varies from 3 seconds to 3 minutes and can be one of these 3 types of transformed fragment of a:

1) reference video.
2) reference video embedded into a video not in the reference video database.
3) a video not in the reference video database.

The total number of audio+video queries for TRECVID 2010 is equal to 11,256 audio+video queries coming from 56 transformations (seven audio × eight video transformations). In TRECVID 2009, 49 audio+video transformations (seven audio × seven video transformations) results in 9,849 audio+video queries.

When we examined the TRECVID 2010 dataset we found other audio transformations not mentioned by NIST. For example, many queries are distorted: small silent segments has been added into query signal (e.g. queries 3353, 3771, and 4200). This complicates the task especially when the query is combined with other transformations such as "mixed with speech". In addition, some queries have undergone time-frequency scale modification by speeding up or slowing down the query (e.g. queries 3030: −180%, 4245: −38%, 3145: −6%, 3056: +8%, 3857: +23%)[2].

We also noticed that many reference files in TRECVID 2010 dataset have duplicates that skew the results. Therefore, we have removed these duplicate files before evaluating any systems we have tested. The duplicate reference files are (removed files are marked in italic): *102*/50, *10907*/5225, *2002*/2062, *3747*/*4328*/9877, *10236*/*11449*/7414, *5680*/7130, *2572*/*2552*/2539, *3725*/5716.

1.2.2 Evaluation metrics

The task in TRECVID CBCD evaluations is to determine for each query if it contains a segment from the reference database. Since the query may be embedded into a non-reference collection (query of type 2), the final result includes the following information when a copy is detected: query start time, reference start time, the reference finish time and confidence value.

[2] These are only approximations of the speed difference between the query and the corresponding reference. For example, -180% means that if we speed up the query by 180% we obtain approximately the speed of the reference.

To evaluate the accuracy of locating a copied fragment within a video, we use Mean F1 (i.e. F-measure) that is defined as the harmonic mean of precision and recall of the detected copy location versus the ground truth location, computed as:

$$Mean\ F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{1.1}$$

where

$$precision = \frac{TP}{TP + FP} \tag{1.2}$$

and

$$recall = \frac{TP}{TP + FN} \tag{1.3}$$

where

- True Positive (TP): the number of positive queries correctly classified by the system.
- True Negative (TN): the number of negative queries correctly classified by the system.
- False Positive (FP): the number of negative queries misclassified by the system.
- False Negative (FN): the number of positive queries misclassified by the system.

To evaluate the detection effectiveness we use the minimal Normalized Detection Cost Rate (min NDCR). NDCR is a weighted cost combination of the probability of missing a true copy (P_{Miss}) and the false alarm rate (R_{FA}), computed as:

$$NDCR = P_{Miss} + \beta \times R_{FA} \tag{1.4}$$

$$P_{Miss} = \mathrm{FN}/N_{target} \tag{1.5}$$

$$R_{FA} = \mathrm{FP}/(T_{refdata} \times T_{query}) \tag{1.6}$$

β is a constant depending on the test condition, and is computed as:

$$\beta = C_{FA}/(C_{Miss} \times R_{target}) \quad (1.7)$$

C_{Miss} and C_{FA} are the costs of an individual miss and an individual false alarm, respectively; FN and FP stands for false negative and false positive, respectively; $T_{refdata}$ is the total length (in hours) of the entire reference dataset, and T_{query} is the total length (in hours) of the queries for a transformation; N_{target} is the total number of copies and R_{target} is the a priori target rate for the application of interest.

In the TRECVID evaluation, different parameters are defined for two different application profiles: "balanced" and "no false alarm" (NOFA) profile. In NOFA profile, which is the most difficult, the cost of an individual false alarm is set to 1000 times the cost of a missed query. In balanced profile both missed and false alarms are equally assigned to 1.

In this work we report results only for the most difficult task which is using the NOFA profile. Thus, the parameters used for the NOFA profile are:

$$R_{target} = 0.005/\text{hr}^2, C_{Miss} = 1 \text{ and } C_{FA} = 1000 \quad (1.8)$$

1.3 Related Work

The area of multimedia copy detection has experienced a surge of research and development activities in the early 2000s. This period has been marked by the amazing growth of the Internet and the emergence of new technologies that have changed the way the multimedia content is used and redistributed. In this context, the multimedia copy detection has become an essential requirement to cope with the illegal distribution of protected multimedia content. Over the years, different issues have been introduced, and the multimedia copy detection has been in continuous evolution allowing the emergence of new solutions.

An important number of multimedia fingerprinting approaches have been published during the last decade. In this section, we review these approaches used to detect audio and/or video transformed copies. First, we describe audio fingerprinting methods that use only the audio signal to detect audio copies. Second, we outline visual feature approaches used to detect video copies. Third, we examine a number of papers using fusion strategy that combine audio and video results. Last, we review the multimedia fingerprinting field from the aspect of types of fingerprints and techniques employed to accelerate the search of audio/video fingerprints.

1.3.1 Audio Fingerprinting

Different audio features for content-based audio copy detection have been used in the past. In a well-known method (Haitsma and Kalker, 2002), the audio signal is segmented into small overlapping frames of 11.6 ms. Then, a *sub-fingerprint* representing a binary code of 32-bits is generated from each frame by computing the energy differences along the frequency and the time axes. Due to the weak discriminative power of a single binary fingerprint, a *fingerprint block* is constructed by collecting 256 adjacent 32-bit sub-fingerprints representing three seconds of the audio signal. This fingerprint block is supposed to contain sufficient data to identify an audio copy.

In order to convert the audio frame into a binary fingerprint, Haitsma and Kalker divide the frequency scale between 300Hz and 2000Hz into 33 non-overlapping frequency bands. Then, the energy differences between these frequency bands are used to compute the sub-fingerprint. Thus, if *E(n, m)* represents the energy of band *m* of frame *n*, and *F(n, m)* represents the m^{th} bit of the sub-fingerprint of frame *n*, then the binary code is computed as follows:

$$F(n,m) = \begin{cases} 1 \; if \; E(n,m) - E(n,m+1) - \big(E(n-1,m) - E(n-1,m+1)\big) > 0 \\ 0 \; if \; E(n,m) - E(n,m+1) - \big(E(n-1,m) - E(n-1,m+1)\big) \leq 0 \end{cases} \quad (1.9)$$

The energy difference fingerprint described above achieved good results when used to identify distorted audio. However, the binary representation of these fingerprints is vulnerable to some audio transformations such as bandwidth limitation or irrelevant speech addition (Saracoglu et al., 2009). An improvement proposed in (Saracoglu et al., 2009) is to reduce the fingerprint from 32 to 15 bits by dividing the audio frame into 16 frequency bands instead of 33 frequency bands. The energy differences between two consecutive frequency bands have been the subject of several other works (Lebosse, Brun and Pailles, 2007; Lezi et al., 2012). In chapter 4, we show that our audio fingerprinting system is more robust to audio transformations compared to the energy difference fingerprint method when tested on TRECVID 2010 dataset.

In the NN-based system (Gupta, Boulianne and Cardinal, 2012), 12 Mel-Frequency Cepstral Coefficients (MFCCs) plus energy and its delta coefficients are used as audio features. The matching algorithm is based on nearest neighbor search that associates the closest query frame to each reference frame. The number of matching frames is then computed between the query and the references to identify the best matching segment. Although the search algorithm is time consuming, this system achieved the best results on TRECVID 2009 evaluation campaign in terms of detection performance.

In order to reduce computing time, Gupta et al. used a fusion strategy to combine the results obtained by the NN-based method and an implementation of the energy difference fingerprint (Haitsma and Kalker, 2002). This allowed them to reduce significantly the run time, while maintaining a good detection performance. In chapter 2 and 4, we compare the performance of our proposed system to the NN-based one.

In another related work, local spectral energies around salient points chosen from the maxima in the Mel-filtered spectra are selected (Anguera, Garzon and Adamek, 2012). Regions around each selected point are encoded to generate binary fingerprints. First, the audio signal is down sampled to 4Khz, and the spectrogram on the range between 300Hz and 3000Hz is generated in a similar way as the method of (Haitsma and Kalker, 2002). Then, a number of

salient points (between 70 to 100 salient points) are selected from the spectral representation by looking for those that have higher energies than all energies in adjacent band-time locations. In order to reduce the number of the selected peaks, a post-detection filtering eliminates those peaks whose energy is below a given threshold. Finally, a binary fingerprint is generated by applying a *mask* centered at each of the selected salient points. The main idea is to compare the average energies between different already defined regions of the spectrogram, so that each compared region pair results in one bit. The final fingerprint is a binary code of 26 bits composed of 22 bits resulting from 23 pair comparisons, and 4 bits that encode the location of the salient peak.

The Mask fingerprinting system (Anguera, Garzon and Adamek, 2012) improved the detection performance by 21.8% compared to the energy difference fingerprint method (Haitsma and Kalker, 2002) on TRECVID 2010 dataset for the balanced profile. We show in chapter 4 that the proposed audio fingerprinting system performs better than the Mask system and give lower min NDCR on TRECVID 2010 dataset even when compared to the difficult no false alarm profile.

The idea of constructing fingerprints based on spectrogram peaks has been used before in the Shazam system (Wang, 2003), where several time-frequency points are chosen from the spectrogram. Shazam starts by looking for spectral points in the same manner as the Mask method. Thus, a point is selected if it has higher energy than all its neighbors in a region centered on the point. The idea is to transform the spectrogram into a "*constellation map"* representing a sparse set of coordinates. Then, a match between an audio copy and its original content is found when a significant number of points coincide between the two constellation maps generated from the copy and the original audio content. However, locating constellation points of the copied audio segment within a large universe of points generated from the original content is time consuming. To overcome this problem, Wang proposed a novel matching approach that accelerates finding constellation points by order of magnitudes.

The solution proposed by Wang to index the constellation map is to transform the latter into fingerprint hashes. To do that, different anchors points are chosen from the selected spectrogram peaks and associated with a fixed rectangular target zone. Then, the frequency components of a pair of points (the anchor and the associated point chosen from the target zone) and the distance between them are hashed into a 32-bits binary code. For example, if (t_1, f_1) are the spectrogram coordinates of the anchor, and (t_2, f_2) are the spectrogram coordinates of the associated point, then the hash is computed as follows:

$$Hash = (f1, f2, t1, t2 - t1) \tag{1.10}$$

Finally, the search is performed by matching the spectrogram points that have the same hashes, and the score between two audio is computed based on a histogram that maintains the number of matched points between the audio segment and each reference. In chapter 2 and 4, we present the performances of Shazam system (Wang, 2003) on TRECVID 2009 and 2010 datasets, and we compare its results to our audio fingerprinting system in terms of detection performance, localization accuracy and search run time.

In (Jegou et al., 2012)**,** an audio fingerprinting system is introduced and offers similar results to Shazam system. They divide the audio signal into overlapping short-term windows of 25 ms taken every 10 ms. Audio descriptors for each window are computed using 64 filter banks (i.e. the dimensionality of one frame is 64). In order to make these descriptors more discriminative, they concatenated 3 successive filter banks. Their scheme resulted in a compound descriptor of dimensionality 192 representing 85 ms. They then use an approximate nearest neighbor search between query descriptors and reference descriptors. This method achieved good results for audio+video copy detection, where these audio descriptors are combined with visual descriptors (Ayari et al., 2011)**.**

Unlike most approaches described above, the authors in (Yan, Hoiem and Sukthankar, 2005) transform the music identification problem into 2-D computer vision problem. They learn a set of filters to create a compact representation for local regions of the spectrogram image.

Thus, a spectrogram is transformed into a set of 32 bit vectors, and a classical hash table is used to perform the search step. Based on (Yan, Hoiem and Sukthankar, 2005), Baluja et al. extend a wavelet-based approach used for near-duplicate image retrieval, to the task of audio detection (Baluja and Covell, 2007). They extract spectral images from the spectrogram and compute Haar wavelets for every image. To reduce the effects of audio degradation, only wavelets with the largest magnitude are selected. A binary representation of the selected wavelets is created, and the Min-Hash technique is used to reduce the size of a fingerprint to a compact representation of *p*-bytes.

The Scale Invariant Feature Transform (SIFT) (Lowe, 2004) technique is used in (Zhang et al., 2015) to extract local image descriptors from the spectrogram. The SIFT algorithm takes the difference of successive Gaussian-Blurred images convolved with Gaussian filters at different scales. Then, a number of key points that represent maxima and minima in the Difference of Gaussian image are selected, and one or more orientations are assigned for each key point. Finally, a 128-dimentional descriptor is generated for each key point based on a set of orientation histograms created around *N* neighborhood of each selected key point. The resulting SIFT features are robust to image stretch and translation. These two characteristics have encouraged Zhang et al. to extract local descriptor from the spectrogram image, which allows them to address Time Scale Modification (TSM) and pitch shifting problems. The evaluation of this system on a dataset containing more than 10,000 of music pieces of various genres showed promising identification results for audio stretched from -30% to +50% and pitch-shifted from -50% to +100%.

1.3.2 Video Fingerprinting

Several kinds of video copy detection methods have been proposed in the literature. A good overview of video fingerprinting techniques can be found in (Law-To et al., 2007; Lu, 2009a). Color-based fingerprints are among the first video features used in video copy detection (Law-To et al., 2007). In (Naphade, Yeung and Yeo, 1999), histograms are used to

represent the distribution of image intensity and color, and the distance between two frames is defined as the histogram intersection.

Another work described in (Shen et al., 2007) combines color histograms with Bounded Coordinate System (BCS) that globally summarizes each video to a single vector allowing a real time processing. In (Wu, Hauptmann and Ngo, 2007), the signatures derived from color histograms are used to detect videos with simple variations. A more expensive detection based on local feature is performed when these features are unable to detect video subjected to complex variations. Despite their popularity, color-based features are sensitive to several video transformations such as insertion of logos, compression in different encoding formats and change of color (Shen et al., 2007).

Other approaches involve using global features such as the ordinal measure introduced originally for computing image correspondence (Bhat and Nayar, 1998). The ordinal measure is used in (Hampapur, Hyun and Bolle, 2002) for the purpose of copy detection. The authors of this method divide each image into *N* blocks and sort them according to their average grey level. The ordinal measure of a given frame is defined by a vector containing the rank of each block. Thus, the ordinal signature of a given frame at time *t* is represented by a vector of integers r_i (i.e. the rank of the block *i*):

$$S(t) = (r1, r2, \dots, rN) \tag{1.11}$$

The distance between the ordinal signature *Q(t)* of the query and the ordinal signature *R(t)* of the reference (of length L_T) is obtained by aligning the query over the reference and computing the distance at different points. The best match between *Q(t)* and *R(t)* at any time *t* corresponds to the alignment where the distance *D(t)* is minimal.

$$D(t) = \frac{1}{L_T} \sum_{i=t-L_T/2}^{t+L_T/2} |R(i) - Q(i)| \tag{1.12}$$

The ordinal measure is used in a similar way in (Chen and Stentiford, 2008). However, instead of ranking regions in the image, a temporal window is used to rank regions along the time scale. The proposed temporal ordinal measure achieved better performance than the spatial signature based on ordinal ranking described in (Hampapur, Hyun and Bolle, 2002), when tested on a small reference database of 3.1 hours.

Another global feature scheme consists of using a Bag-of-global visual feature based on a DCT-sign-based feature (Uchida, Takagi and Sakazawa, 2011). The authors performed multiple assignments of visual words in the feature, spatial, and temporal domains to improve repeatability of Visual-Words based feature matching. Their system processes queries 60 times faster than real time and results in good performance on TRECVID datasets for non-geometric video transformations.

In (Esmaeili, Fatourechi and Ward, 2011a), TIRI-DCT is proposed to generate spatio-temporal fingerprints based on the Temporally Informative Representative Images (TIRI) method introduced in (Malekesmaeili, Fatourechi and Ward, 2009). Instead of extracting visual feature from each video image, this method segments the video into shorter clips and combines all the frames in each video sequence to generate one representative image. Then, each TIRI image is transformed into a hash vector representing the final video fingerprint.

In a more recent work (Nie et al., 2014), the video is modeled using a graph that represents the relations among different frames, and the weights of the graph are calculated using a structural similarity algorithm. This graph-based modeling scheme achieves good results when evaluated in a small dataset, and performs as well as the TIRI-based method (Esmaeili, Fatourechi and Ward, 2011a).

On the other hand, several papers propose to generate fingerprints based on local information of the image (Douze, Jegou and Schmid, 2010; Heritier et al., 2009; Liu, Liu and Shahraray, 2009; Yeh, Hsu and Lu, 2010). Local features have shown their robustness against content-changing transformations compared to global features. A comparison between global feature

(based on the ordinal feature) and local features (SURF and HOOF features) shows that the local features outperform global features when evaluated on three different datasets (Chou, Chen and Lee, 2015). Similarly in (Chiu et al., 2014), local features based on SIFT show their robustness against transformations that change the content of the video frame compared to the global based feature.

1.3.3 Audio+Video Systems

Although a large number of the proposed methods focus on either audio or video fingerprints, some are based on multimodal features, where the audio and visual information are used to detect video copies.

A good multimodal feature representation using complementary audio features, local visual features and global visual features is described in (Mou et al., 2013). The audio part of this system is based on the Weighted Audio Spectrum Flatness (WASF) features introduced in (Chen and Huang, 2008). These features extend the MPEG-7 descriptor by introducing the Human Auditory System (HAS) functions to weight the audio signal. Mou et al. extract 14-dimentional WASF features from each audio frame of size 60 ms. Then, they combine the WASF features generated from 198 audio frames and reduce them to a vector of 72-dimentions using a technique specified in MPEG standard (Carpentier, 2005). The resulting 72-dimentions vector represents the signature of four seconds length audio clip.

For their video part, a local visual feature of dense color SIFT (DC-SIFT) (Bosch, Zisserman and Munoz, 2008) is used as local feature, whereas the global visual feature is based on DCT feature (similar to the DCT introduced in (Ching-Yung and Shih-Fu, 2001)). The similarity search is performed using a temporal pyramid-matching algorithm, where several techniques are employed to speed up the search. Locality Sensitive Hashing (LSH) (Indyk and Motwani, 1998) technique is used to index DCT and WASF features, and a bag of words is applied to convert each DC-SIFT vector into a visual word that is stored in an inverted index.

This system achieves excellent results on TRECVID 2009 and 2010 datasets when the results obtained separately by these features are combined by applying a result-level fusion mechanism (Mou et al., 2013). In chapter 3, we compare the performance of our video fingerprinting system to the DC-SIFT and DCT used by this system.

In (Li et al., 2014), binary audio fingerprints generated with the energy difference fingerprint method (Haitsma and Kalker, 2002) are used with the ordinal signature proposed in (Bhat and Nayar, 1998) as visual feature to perform a two-step search. In the first step, a number of candidate videos are selected based on the audio only results. Then in the second step, visual features are extracted from the selected candidates and combined with the audio fingerprints to produce the final results.

In (Mukai et al., 2010), the spectrogram of an audio is divided into small regions and then quantized by Vector Quantization (VQ). A VQ codebook is prepared for each frequency band, and the similarity search is performed based on the VQ codes and an index list. For the video part, Mukai et al. used the Coarsely-quantized Area Matching (CAM) algorithm that extracts global features from video images by looking for salient parts of frames. A salient part is an image area that presents the highest RGB values changes over a time window. These global features are used in (Gupta et al., 2012) with a nearest-neighbor mapping technique that gives better performance.

The nearest-neighbor mapping technique is also used for their audio copy detection system, where MFCCs are used as audio features (Gupta, Boulianne and Cardinal, 2012). This system achieved very good results for audio+video copy detection task when tested on TRECVID 2011 dataset. It achieved the best performance across all sites for 25 out of 56 transformations (Gupta et al., 2012). In chapter 3, we compare our video fingerprinting approach to this system using TRECVID 2009 and 2010 datasets.

1.3.4 Accelerating Fingerprints Search

As stated above, a typical content-based copy detection system is mainly composed of two parts: feature extraction that converts the audio signal into a set of fingerprints, and a method to search for the extracted query fingerprints in the reference database. The robustness and speed of the CBCD system depends on the reliability of these two components. We present in this section a review of previous approaches with special attention to the impact of the fingerprint types and the search algorithm on speed.

The energy difference based fingerprint (Haitsma and Kalker, 2002; Lebosse, Brun and Pailles, 2007; Lezi et al., 2012; Saracoglu et al., 2009), where the fingerprint is represented by a binary code, figures among the fastest audio fingerprinting systems. The Hamming distance is used to compute the number of bit errors between two fingerprints. However, instead of comparing each query fingerprint to all reference fingerprints, the calculation is limited to a few candidate positions by using a lookup table, allowing a very fast search. The binary representation of the fingerprints makes the search very fast. However, this fast search results in a modest performance compared to other methods. Several other approaches based on binary fingerprints have been proposed (Anguera, Garzon and Adamek, 2012) (Yan, Hoiem and Sukthankar, 2005), and have shown significant improvements over the energy difference based fingerprints.

The Shazam system (Wang, 2003) is another good example of very fast system where fingerprints are transformed into hashes for an efficient search. First, the system converts the reference and the query spectrograms into constellation maps where each point within a map denotes a time-frequency peak. A combinatorial hashing technique is then used to transform the number of peak points (anchors) into hashes. Each anchor is then paired with points within its target zone, and each pair generates a fingerprint that encodes two frequency components and the time difference between the pair of points. This technique allows a fast lookup search in constant time instead of matching individually each point on the constellation map. Although very fact, the detection performance of Shazam is relatively

poor compared to other system that do not use binary fingerprints when evaluated on TRECVID 2009 and TRECVID 2010 datasets (see chapter 2 and chapter 4 for the evaluation results on these datasets).

On the other hand, approximate searching techniques such as locality-sensitive hashing are used in several works to accelerate the search. In (Baluja and Covell, 2007; Baluja and Covell, 2008), wavelets with the largest magnitude are selected from the spectrogram, and locality-sensitive hashing technique is used to accelerate the fingerprint similarity search. Although more robust than the energy difference fingerprint method (Haitsma and Kalker, 2002) and Shazam (Wang, 2003), this system is computationally very expensive (Anguera, Garzon and Adamek, 2012). A comparative study of (Yan, Hoiem and Sukthankar, 2005), (Wang, 2003) and (Baluja and Covell, 2007) in a common framework in terms of detection accuracy and computation time can be found in (Chandrasekhar, Sharifi and Ross, 2011). In (Mou et al., 2013), the WASF is used as audio features, and LSH is adopted to compute the dissimilarity between two WASF features. locality-sensitive hashing is used in many other works to accelerate the search, but it is slower than the hashing-based search (Yan, Hoiem and Sukthankar, 2005).

Recently, Graphics Processing Units (GPUs) have been used as a powerful way to accelerate scientific computations. Using GPU to accelerate large-scale applications became easier with the Compute Unified Device Architecture (CUDA) platform introduced by NVIDIA. Several GPU implementations of widely used algorithms such as k-nearest neighbor (Garcia, Debreuve and Barlaud, 2008) and LSH (Pan and Manocha, 2011) have been proposed and can be adopted for audio copy detection systems. A GPU implementation of the Metric Permutation Table algorithm is proposed in (Mohamed, Osipyan and Marchand-Maillet, 2014) to speed up the search of digital images**.**

In (Gupta, Boulianne and Cardinal, 2012), a GPU implementation of the NN-based system is proposed to perform the nearest neighbor search between reference and query frames. The GPU implementation of this approach is described in (Cardinal, Gupta and Boulianne, 2010),

where the copy detection algorithm has been adopted to perform advertisement detection. Compared to its CPU implementation, this GPU implementation improves speed of the search algorithm by a factor of 70.

A parallel implementation of Shazam (Wang, 2003) is introduced in (Wang, Jang and Liou, 2014) and tested over a large database of more than 11,600 hours of audio. A GPU is used to parallelize two parts of the system leading to an overall speedup of a factor of 5. The authors also explored the use of three GPUs instead of only one allowing them to further improve the performance by a factor of three on some parts of the system.

In another work, the computation of the cross-correlation between two audio windows is accelerated using a GPU (Martinez et al., 2011). However, the database used to test the algorithm is very small (one hour), and the GPU lead to a moderate improvement of a factor of two compared to the CPU implementation.

Another technique that aims to reduce search time and complexity of the system described in (Haitsma and Kalker, 2002), is proposed in (Bellettini and Mazzini, 2010). Starting from the fact that the search time is related to the size of the database, the authors partitioned the database of 100,000 songs into ten sub-databases. The search is then divided into ten independent processes executed in different machines. Similar to (Bellettini and Mazzini, 2010), the authors of (Sui, Ruan and Xiao, 2014) divide the fingerprints database into several parts, and the search algorithm is executed in parallel based on the Message Passing Interface (MPI) standard.

On the other hand, clustering based techniques have been used in several works with the aim of avoiding exhaustive search. In (Esmaeili, Fatourechi and Ward, 2011b), binary fingerprints of the reference videos are grouped into k different clusters, and only fingerprints that belong to the cluster closest to the query fingerprint are searched to find a match. The algorithm continues to examine other clusters in the same manner if a match is not found.

The problem in using this strategy is the possibility of visiting all the k clusters before a match is found resulting in an exhaustive search.

In another related work, a bag of audio words model is proposed to group MFCCs and RASTA-PLP (Hermansky and Morgan, 1994) features into different feature spaces (Liu et al., 2010). This model combined with inverted file retrieval makes large-scale audio copy detection possible in real time. However, the detection performance is disappointing especially when queries are distorted by adding irrelevant speech. The technique used in (Liu et al., 2010) is inspired by the bag of visual words which is applied in several video copy detection systems (Douze et al., 2008; Mou et al., 2013; Sivic and Zisserman, 2003; Younessian et al., 2010).

1.4 Summary

In this chapter, we have presented a review of the multimedia copy detection field. This chapter was divided into three parts. The aim of the first part is to give the reader an overview of the area of content-based multimedia copy detection. Several aspects of a multimedia fingerprinting system have been presented, such as the basic structure and fundamental requirements of a multimedia copy detection system based on fingerprints. We also provided some examples of real-word applications that rely on the fingerprinting technology.

The second part of this chapter presented the framework evaluation that has been used during our work. This framework includes the well-known TRECVID 2009 and 2010 copy detection datasets provided by NIST, and all metrics used to evaluate the detection performances of a multimedia copy detection system. These two datasets include video and audio data, and will be used for all the experiments presented in the next three chapters.

The last section of this chapter presented a thorough overview of the related work. More precisely, this section covered previous approaches that use audio fingerprints, video

fingerprints or both types of fingerprints to detect copies. In addition, several techniques described in literature to accelerate the search of fingerprints have been presented, such as the clustering technique or the use of GPU.

CHAPTER 2

AUDIO FINGERPRINTING

In this chapter, we describe our audio fingerprinting system. We propose a novel approach to extract audio features from the spectrogram that allows the generation of three different fingerprints: Global Mean, Local Mean and Salient-Regions. We also describe the search algorithm that performs the match between query fingerprints and reference fingerprints. This algorithm is suitable for audio and video fingerprints retrieval. We evaluate the performance of the proposed fingerprint extraction methods in terms of detection performance, localization accuracy, and processing run time using TRECVID 2009 and 2010 datasets. In the last part of this chapter, we compare our system performance with two state-of-the-art audio copy detection systems.

2.1 System Overview

The overall architecture of our system is shown in Figure 2.1. First, spectrogram-based *d*-dimensional vectors per frame are generated from all the audio references. A spectrogram is a representation of the frequency spectrum of a sound (y-axis) as they vary with time (x-axis). Second, different versions of these *d*-dimensional vectors per frame are created (by varying the spectral energy threshold) and stored in reference fingerprints database. A query is processed in the same way, and followed by a time-frequency shift step in order to produce several fingerprint versions with different speeds. Finally, the query fingerprints are searched in the reference fingerprints database to produce the search results. In the following paragraphs we describe each step in more details.

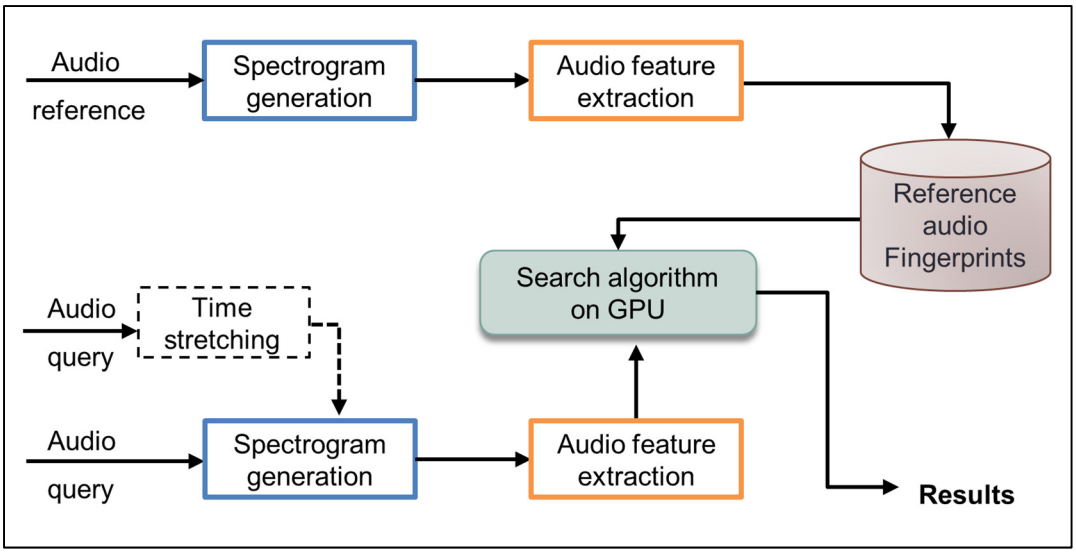

Figure 2.1 Proposed audio fingerprinting system overview

2.2 Spectrogram Generation

First, we down sample the audio signal to 8 KHz in order to make the fingerprint robust to transformations that may reduce the bandwidth of the audio signal. We apply Hamming window of 96 ms duration. We generate a spectrogram by computing the short-time Fourier transform in this 96 ms window. We reduce this spectrogram to 257 frequency bins in the frequency range from 500 Hz to 3000 Hz. We compute these 257 frequency bins every 3 ms.

2.3 Fingerprint Generation

The previous step transforms the audio into a spectrogram represented by a matrix containing the energy of the signal at any given time and frequency. The fingerprint generation step converts this spectrogram matrix into binary images using a sliding window of size $w \times h$. These images are then converted into multiple d-dimensional vectors as outlined below.

The spectrogram matrix of size $w \times h$ is converted into multiple d-dimensional vectors using varying spectral thresholds. We compute these d-dimensional vectors every av ms (frame advance of size av). The choice of these parameters is discussed in Section 2.6.1.

We propose two different audio features derived using the global or local mean of the spectral values in the spectrogram: Global Mean and Local Mean fingerprints. In addition, we propose a third fingerprint extraction scheme that outperforms Global Mean and Local Mean fingerprints. We describe in the next sub-sections these three fingerprint extraction methods. Then, we will present the performance of each method in section 2.6.

2.3.1 Global Mean Fingerprint

Figure 2.2 illustrates the Global Mean fingerprint generation process. From the spectrogram matrix of size $w \times h$ (i.e. the window frame), we compute the mean energy value of this matrix. Then, we replace the energy values of this matrix by either 0 or 1 as follows: if the intensity is greater than this global mean then we replace it by 1, otherwise we replace it with a 0. Thus, the global mean represents a threshold that generates one fingerprint version.

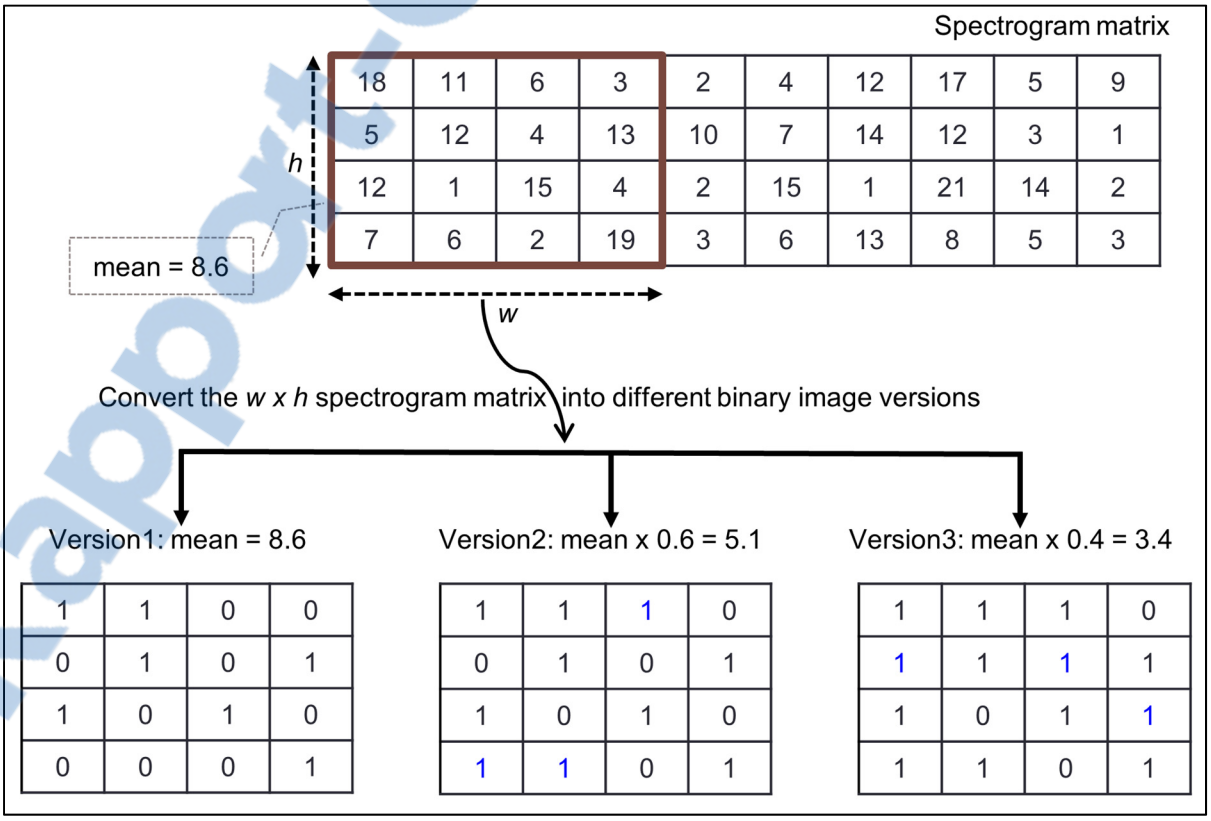

Figure 2.2 Global Mean fingerprint extraction

We generate different versions of the fingerprint from the same spectrogram matrix by using different thresholds derived from this global mean (e.g. 0.4 × global mean, 0.6 × global mean). Figure 2.3 shows four versions generated for two reference frames and two query frames. Note that images in each line are not successive frames, but different version of the same frame differing in spectral threshold (or global mean) for quantization. In this figure, *query1* is a transformed copy (bandwidth limit and single-band companding) of *reference1*, and *query2* is a copy of *reference2* mixed with speech. Note how easy it is to match *query1 version3* to *reference1 version2*. Similarly, *query2 version4* and *reference2 version4* are almost identical despite the fact that *query2* is mixed with extraneous speech.

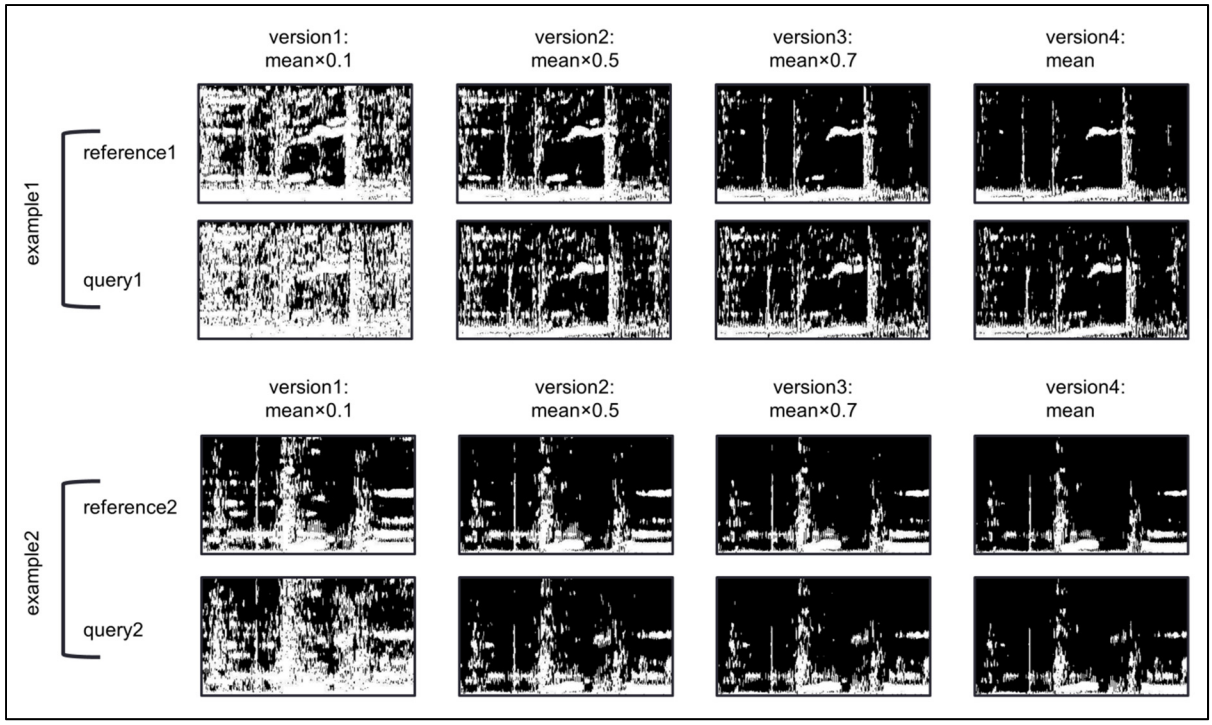

Figure 2.3 Four different versions of quantized spectrogram matrix generated for four different audio frames (1-sec window frame)

This scheme results in robust fingerprints against a variety of transformations. Using the mean of energies as a threshold allows us to select the most relevant information and discard the irrelevant information, for example, eliminating low energy noise from fingerprint representation. In addition, the binary representation makes the fingerprint invariant to

relative energy value changes (e.g. overall amplification or reduction of energy, equalization…). Thus, a value of 1 in the $w \times h$ binary spectrogram matrix (i.e. the binary image) denotes a time-frequency peak regardless of the real intensity value. This is similar to the Shazam feature extraction (Haitsma and Kalker, 2002) where the amplitude component has been eliminated and only peak positions are retained.

2.3.2 Local Mean Fingerprint

Unlike Global Mean feature, Local Mean uses smaller blocks to compute the mean. In fact, we partitioned the $w \times h$ window into tiles of $w_b \times h_b$ each (see Figure 2.4). We then compute the mean energy values of each tile block, and convert the energy values in each tile to a 0 or 1 following the strategy used for Global Mean.

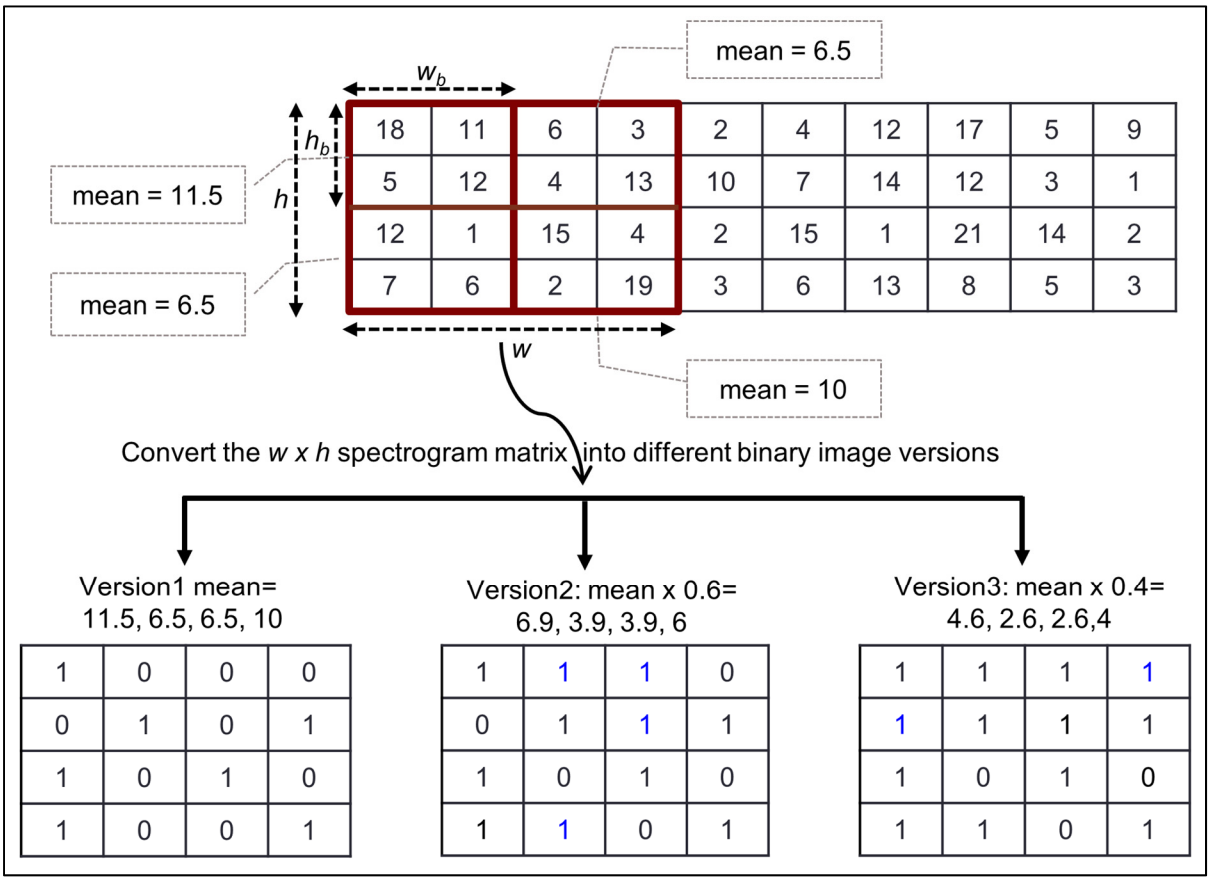

Figure 2.4 Local Mean feature extraction

Figure 2.5 shows binary images of the same audio segment (1-sec length) generated with Global Mean and Local Mean using the threshold equal to *mean*. The query in this figure is a transformed version (with *mp3 compression and multiband companding*) of the reference. We compare Global Mean which extracts "the most relevant information" on the whole $w \times h$ window, to Local Mean that emphasizes the intensity values of a small portion of the window ($w_b \times h_b$). In other words, Local Mean represents not only the highest energy values in the window like Global Mean but also smaller local energy values that are not relevant in the case of the Global Mean. These smaller local energies can be useful in minimizing the difference between query and reference images. In Figure 2.5, we can see that Local Mean can generate images for the reference and the query that are more similar than those generated by the Global Mean.

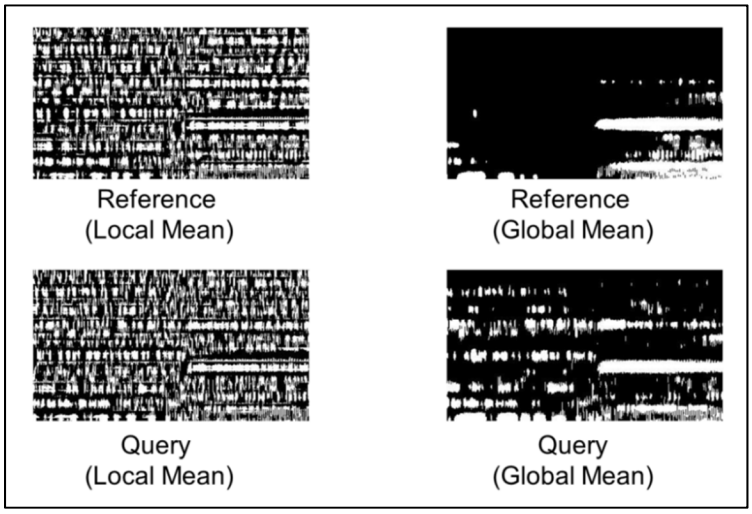

Figure 2.5 Global Mean versus Local Mean Fingerprints

We represent the resulting binary image, generated either by Global Mean or Local Mean, (or the $w \times h$ spectrogram matrix if the spectrogram is not converted into binary images) using a simple d-dimensional fingerprint. We divide the matrix into $d/2$ horizontal slices and $d/2$ vertical slices. We then take the sum of the elements of each slice to obtain a vector of d dimensions (see the example given in Figure 2.6 with $d = 48$). This d-dimensional vector is the compact fingerprint representation of the $w \times h$ spectrogram matrix.

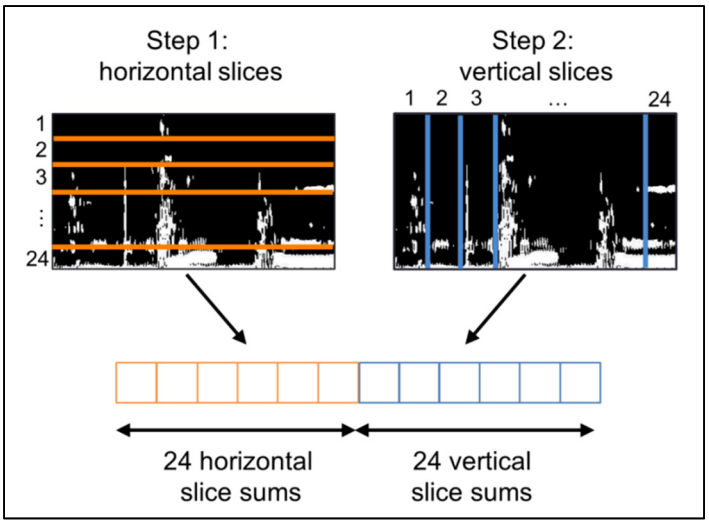

Figure 2.6 Fingerprint representation of Global Mean and Local Mean fingerprints

2.3.3 Salient-Regions Fingerprint

The first step of the Salient-Regions fingerprint extraction is similar to the Global Mean fingerprint, where the audio signal is transformed into binary images by using the mean of the $w \times h$ window as threshold. The main difference between these two fingerprints is in the fingerprint representation, as explained below. Besides, with Global Mean different fingerprint versions are generated from the same spectrogram matrix by using different thresholds. Generating different versions of fingerprints improves likelihood of one of the query fingerprint version matching a reference fingerprint version. However, this technique has a direct impact on the processing run time, which is multiplied by the number of these fingerprint versions. Thus, we propose to generate only one fingerprint version (using threshold = *mean* of the $w \times h$ window) with the Salient-Regions method.

In order to improve the fingerprint representation used with Global Mean and Local Mean that divide the binary image using horizontal and vertical slices, we propose to divide the binary image into small square tiles of a fixed size, so each element of the d-dimensional vector is the sum of a small square. This strategy should results in a more accurate representation of the image for two reasons. First, in the previous image representation used

with Global Mean and Local Mean, we process each region of the image twice (one to compute the vertical slice and one for the horizontal slice). In this previous scenario, if a noise was introduced in the image, then two elements of the d-dimensional vector were affected. Here, only one element of the tile-based representation is affected. Secondly, when using horizontal and vertical slices, an image can be divided into a maximum of 590 slices (257 horizontal slices + 333 vertical slices) compared to 85581 (257 × 333) regions with tile-based representation (when using 1 × 1 tile).

In this work we use a tile of size 11 × 11, which results in D = 744 tiles. However, a 744 dimensional vector is very large, increasing considerably the search processing time. Therefore, we select only a few salient tiles and discard the rest of the image. We select d tiles that represent the highest values. To encode the fingerprint, we keep only the position of the selected tile and eliminate its value. In other words, we divide the image into 744 tiles and we compute the sum in each tile. We number each tile of the image from 1 to 744, and then we look for the d tiles that have the highest values. The fingerprint represents the positions of these d tiles. Figure 2.7 illustrates this binary image feature extraction step. This d-dimensional vector is the compact fingerprint of the $w \times h$ binary spectrogram matrix.

This approach differs from methods like MASK (Anguera, Garzon and Adamek, 2012) or Shazam (Wang, 2003) where salient points are selected directly from the spectrogram. In our work, features are extracted from the binary image that describes the shape of the signal after noise suppression and elimination of signal amplitude. We demonstrate in Section 2.6.4 that the resulting binary images are more robust to audio distortions compared to directly extracting features from the spectrogram (i.e. without noise suppression). We reduce the search time by quantizing the binary image and selecting the most relevant regions. We believe that salient regions are more robust than salient points, especially for transformations that add irrelevant speech to the signal. Different regions of the binary image are less likely to be distorted than different points in the spectrogram.

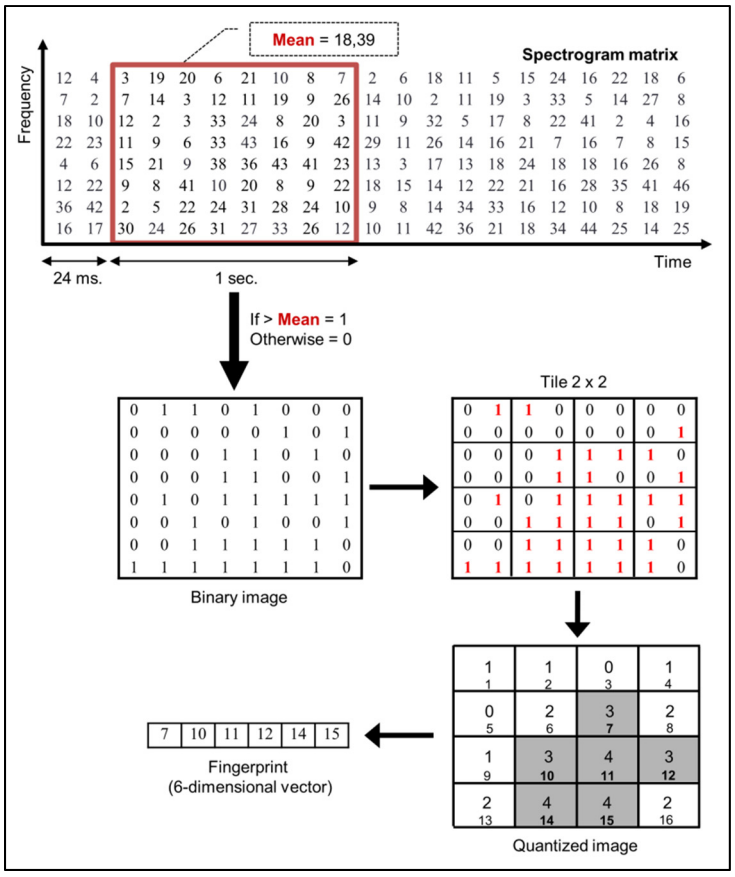

Figure 2.7 Feature extraction with 16 tiles (D = 16) and 6 salient tiles (d = 6)

2.4 Query Fingerprint Generation

We create query fingerprints in the same way as reference fingerprints. However, we noticed frequency sampling modification (i.e. time-frequency scale modifications) between some queries and their references (only on TRECVID 2010 dataset). Note that changing the frequency sampling implies changing the duration and the frequency content of the signal. To overcome these differences, we also produce query fingerprints that have been speeded up or slowed down by 9%. We do that by increasing/decreasing the sampling frequency of the audio file (i.e. time-frequency scaling by resampling).

This parameter (i.e. 9%) is chosen to detect audio copy that has undergone a small time-frequency scale variation. However, the problem becomes more difficult when the time-frequency scale difference between the query and the reference is large. We found that TRECVID 2010 contains many queries that represent time-frequency scale differences compared to their corresponding references (see Section 1.2.1). Our experiments show that changing the speed by ± 9% does not detect all these transformed copies. For TRECVID 2009 dataset we did not generate additional fingerprints as this dataset has not undergone any time-frequency scale variation.

Note that time and/or frequency scale modifications have a large impact on detection performance for many CBCD systems (Baluja and Covell, 2008). Different approaches have been proposed to handle this specific kind of audio transformations (Malekesmaeili and Ward, 2014; Ramona and Peeters, 2013; Zhang et al., 2015).

2.5 Retrieval

Once all the reference and query fingerprints have been created, a search is performed to see if the query is a copy of an original audio in the reference fingerprint database. This search is an adaptation of the search algorithm introduced in (Gupta, Boulianne and Cardinal, 2012), and it is composed of two principal steps. In the first step, each reference fingerprint is labeled with the frame number of its closest query fingerprint. To find the frame number of the closest query fingerprint, a similarity search algorithm is performed between the reference fingerprint and the query fingerprints. In the second step, we compute the number of matching frames between the query and the reference frames. The next subsections describe in detail this algorithm.

2.5.1 Similarity Search

During retrieval, each query fingerprint version is compared with all the reference fingerprint versions (only one version with Salient-Regions fingerprint is used). Ideally, reference audio

frames and its near duplicate queries should have identical fingerprints among these versions. However, even when images look identical, their *d*-dimensional descriptors could differ slightly. Furthermore, distortions, such as "mixed with speech", can make images very different. Thus, a natural choice is to use a similarity measure between two fingerprints that is robust to audio distortions. Depending on the fingerprint representation, one of the two similarity measures described below is used.

2.5.1.1 Similarity Measure for Global Mean and Local Mean Fingerprints

Each generated fingerprint by either Global Mean or Local Mean fingerprints is represented by a *d*-dimensional vector, where each element of this vector is the sum of one slice of the binary image. To find the closest query frame, we use the nearest neighbor algorithm with the Manhattan distance (i.e. absolute distance) as a measure of similarity. Formally, if $X = \{x_1, ..., x_d\}$ and $Y = \{y_1, ..., y_d\}$ then the distance between X and Y is obtained by:

$$\text{Distance (X, Y)} = \sum_{i=1}^{d} |Xi - Yi| \tag{2.1}$$

2.5.1.2 Similarity Measure for Salient-Regions Fingerprint

Salient-Regions fingerprints encode the *positions* of salient regions of the binary images. Thus, the similarity between two fingerprints is defined as the intersection between the elements of these two fingerprints. For example, if $F1 = \{1, 3, 4, 6, 8\}$ and $F2 = \{1, 2, 4, 7, 9\}$, then the *similarity (F1, F2) = count {1, 4} = 2*. Formally, the similarity between *F1* and *F2* is defined as:

$$Count\ (F1 \cap F2) = \text{count}\{x: x \in F1 \text{ and } x \in F2\} \tag{2.2}$$

These similarity computations constitute over 99% of the search time. To accelerate these similarity computations, we implemented two different similarity search algorithms on the GPU in order to compare their execution times. Chapter 4 describes in detail GPU

implementations of these two similarity search algorithms used for Salient-Regions fingerprint retrieval.

2.5.2 Matching

After the closest query frame has been found for each reference frame, the total number of reference frames that match the query frame-synchronously is computed as follow: we move the query over the reference, and we count the number of reference frames that match exactly the query frame number for each alignment of the query to the reference. This count represents the confidence in the match between the query and the reference (i.e. the score). Note that this step is performed in the same manner regardless of the similarity measure used to computer the distance between two fingerprints. Figure 2.8(a) shows a representation of this technique that compute the number of matching frames between a query and a reference.

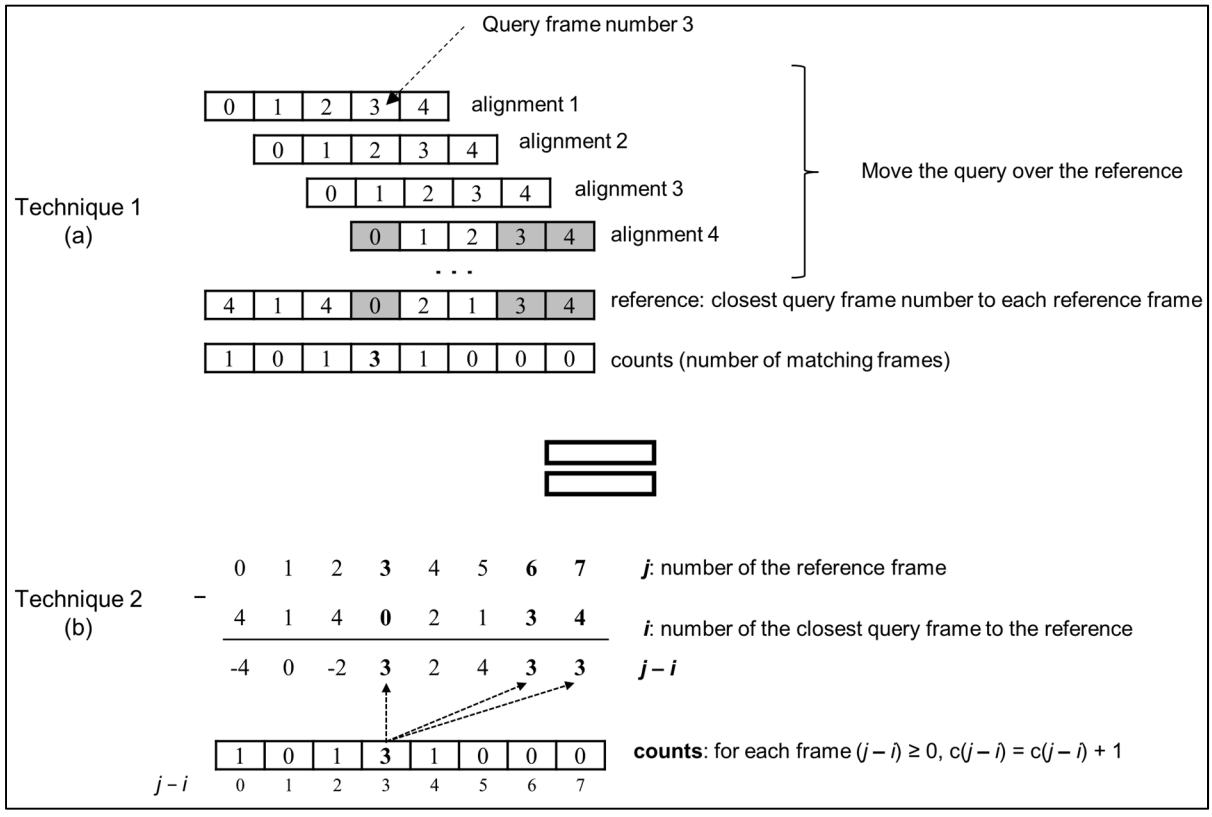

Figure 2.8 Matching frames based on nearest neighbor search

In Figure 2.8(a), the reference fingerprints are labeled with the frame number of the closest query fingerprint (this operation is performed in the previous step: similarity search). The row labeled "counts" shows the total number of matching frames for each alignment of the query to the reference. Then, the best segment match is found by looking for the reference frame with the highest count. In the example of Figure 2.8(a), the best count is equal to 3 and is obtained with *alignment 4* (matched frames between the query and the reference are highlighted with grey background).

A faster algorithm to perform this same operation is proposed in (Gupta, Boulianne and Cardinal, 2012), and illustrated in Figure 2.8(b). The counts of matching frames using this algorithm are obtained as follows: for each reference frame j, we increment the count $c(j - i)$ as follow:

$$c(j - i) = c(j - i) + 1 \tag{2.3}$$

where:

- j is the frame number of the reference.
- i is the label of frame j (i.e. the frame number of the closest query fingerprint is i).
- $(j - i)$ is the index of the vector labeled *counts*.
- $c(j - i)$ corresponds to the contents of the vector labeled *counts* at position $(j - i)$.

In Figure 2.8(b), the best matching segment is obtained when the first query frame is aligned with the reference frame number 3 and results in 3 matching frames. These 3 frames are: (j = 3, i = 0), (j = 6, i = 3) and (j = 7, i = 4).

As we can see from Figure 2.8(b), the total computing for the *matching* step is proportional to the total number of reference frames m, while the computing for the *similarity* search is proportional to $n \times m \times d$. In fact, the *matching* step is very fast and the total search time is dictated by the similarity search time.

Our algorithm differs from (Gupta, Boulianne and Cardinal, 2012) as follows. For each reference frame, we associate the nearest query frame and the N nearest successive neighboring frames of the query. For example, in Figure 2.9 (SCF-1: successive closest frames with $N = 1$), the closest query frame to the fifth reference frame is frame 2. Thus, we update the count $c(j-i)$ for all query frames: 1, 2 and 3 (frames 1 and 3 are successive neighboring frames). We process in this manner because large overlap between frames generates similar fingerprints for successive frames. This similarity allows the search algorithm to label the reference frame with a wrong query frame number. For example, if the correct closest query frame to the reference frame is frame n, then the matching algorithm may wrongfully label frame $n - 1$ as its closest frame since query frames n and $n - 1$ are similar due to the large overlap (24 ms frame advance and one second window). We discuss in detail the influence of the parameter SCF (number of successive closest frames) in Section 2.6.1.

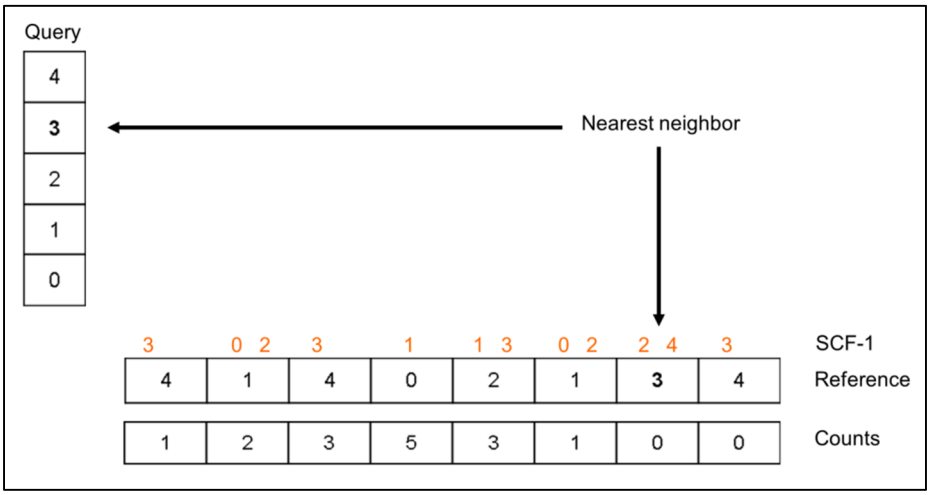

Figure 2.9 Improved matching frame algorithm

2.6 Results and Analysis

This section covers an evaluation of our proposed fingerprints extraction approaches on TRECVID 2009 and 2010 audio copy detection datasets. First, we present results of Local Mean (LM) and Global Mean (GM) fingerprints separately. We then show that the

combination of these two features results in significantly lower min NDCR. We also study the influence of the number of successive closest frames on the system performance. In addition, we show the effectiveness of converting the spectrogram matrix into a set of binary images and the utility of using different fingerprint versions. Besides, we study the performance of our system with the Salient-Regions (SR) fingerprint, and we show that this fingerprint outperforms Global Mean and Local Mean fingerprints. Finally, we compare our results obtained with Salient-Regions fingerprint to the NN-based (Gupta, Boulianne and Cardinal, 2012) and Shazam (Haitsma and Kalker, 2002) audio fingerprinting systems. For the Shazam system, we used the Shazam implementation found in (Ellis, 2009) with the default parameters.

2.6.1 Results for Global Mean Fingerprint

We created four different Global Mean fingerprint version for each query and reference audio file by varying the threshold based on the global mean. The thresholds used to create these versions are: *global mean of the* $w \times h$ *matrix*, $0.6 \times$ *global mean*, $0.4 \times$ *global mean* and $0.2 \times$ *global mean*. As we reduce the threshold, each subsequent fingerprint version includes more spectral information than the previous one (there are more non-zero values in the resulting $w \times h$ binary image). Thus, image generated with the threshold equal to the *global mean* contains less information than the image generated from a threshold of $0.6 \times$ *global mean*, and so on (see Figure 2.3). To increase the likelihood that one query fingerprint version matches a reference fingerprint version, it is better to generate many versions with varying thresholds. However, the run time increases quadratically with the number of versions, leading us to use only a few versions.

In order to study the distribution of the quantized data (number of 1's contained in the binary image) per version, we represent in Table 2.1 the percentage of 1's in the binary images averaged over all reference/query frames for each threshold described above.

Table 2.1 Percentage of 1's in the binary images averaged over all query/reference frames for Global Mean fingerprints

Threshold	0.2 × mean	0.4 × mean	0.6 × mean	1 × mean
Query version (%)	41.9	30.9	24.8	17.9
Reference version (%)	25.5	15.6	13.8	8.9

From Table 2.1 we can see that the 1's are quite sparse at the threshold of *1 × mean*. Therefore, thresholds above this value are not necessary, since higher values will have less discriminative information; which may lead to more false matches between the query and the reference frames. On the other hand, using threshold = *0.2 × mean* generates almost 3 times more 1's, which seems to be sufficient for including additional discriminative binary images. Note that the data generated with this smallest threshold has 1's in 41.9% of the query binary bins, and using a threshold below *0.2 × mean* would only increase the noise especially when the query includes extraneous speech, making the copy detection more difficult. Thus, the range from *0.2 × mean* to *1 × mean* seems sufficient to cover the significant spectral regions.

Based on preliminary tests, we found that four thresholds gave good results on the TRECVID 2009 and 2010 datasets. In addition to these four fingerprint versions for the reference and the query, we generated eight more fingerprints for the query: four with 9% slower audio and four with 9% faster audio.

As stated above, the spectrogram matrix obtained from the audio signal is converted into a set of 2-D binary images. Each image is a quantized version of the spectrogram matrix of size $w \times h$. In our experiment, we use a window of 1-sec length ($w = 333$ and $h = 257$; which corresponds to 333 frames × 257 frequency bins) taken every 24 ms (i.e. $av = 24$). The choice of 1-sec window size ensures that the frame contains enough information to be discriminative. The 24-ms frame advance is a compromise between having too many reference frames being the same (frame advance too short) or missing the alignment between the query and the reference (frame advance too long).

In fact, the choice of the frame advance, when generating fingerprints, impacts the system performance. Short frame advance generates many successive frames with identical fingerprints, especially when the spectral threshold is high (equal to the 1-sec spectrogram matrix *mean)*. This leads to many more false fingerprint matches during search. A larger frame advance avoids this problem. However, a large frame advance can cause problems in matching a reference frame to a query frame. This is because the start of the query may not be synchronized with the start of the reference leading to many more poor matches. The step size of 24 ms chosen in our experiments seems to be a good compromise. Increasing frame advance beyond 24 ms increases min NDCR. However, when we examine our results in-depth, we notice that wrong fingerprint matches still cause many false alarms, especially for short queries. Our use of coarse fingerprints amplifies this problem. A solution to this problem is to increase the dimension of these fingerprints, so the fingerprint will include more details about the image.

To confirm this reasoning, Table 2.2 compares min NDCR and the number of missed queries for Global Mean fingerprint using 26 dimensions versus 48 dimensions for the seven audio transformations: (T1) nothing, (T2) mp3 compression, (T3) mp3 compression and multiband companding, (T4) bandwidth limit and single band companding, (T5) mix with speech, (T6) mix with speech, then multiband compress, (T7) bandpass filter, mix with speech and compress. As we can see from this table, the min NDCR is reduced for all transformations when we represent the binary image by 48 dimensions instead of 26 with a relative improvement of 21%. Similarly, the total number of missed queries is reduced by 17% when using 48 dimensions instead of 26 dimensions.

Table 2.2 Performance of Global Mean with varying dimensions for different transforms

	Dimension	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**
Min NDCR	**26**	0.097	0.104	0.216	0.194	0.313	0.425	0.328
	48	0.09	0.097	0.179	0.134	0.209	0.336	0.284
Miss count	**26**	10	10	25	20	24	48	31
	48	10	10	17	17	24	35	26

As expected, the performance of our system decreases from transformation T1 to T7 in term of miss count and min NDCR. This difference in performance is especially noteworthy for the last three transformations. This is because these transformations (T5, T6 and T7) add irrelevant speech to the query, which makes them more difficult to match. Indeed, if we compare the average min NDCR of transformations that do not add irrelevant speech, to those that add irrelevant speech to the queries, we find that the average min NDCR goes up from 0.125 to 0.276 with 48 dimensions and from 0.152 to 0.355 when we use 26 dimensions.

Note that the missed queries for T1 and T2 are either distorted with large time-frequency scale shift (e.g. +23%, −180%, −38%, etc.), or the query consists only of silence (e.g. queries 3524 and 4315). The presence of silent audio queries can be explained by the fact that TRECVID 2010 dataset was designed to evaluate combined audio+video copy detection. There was no separate audio only copy detection evaluation. In addition to these queries, a large number of missed queries are short queries (less than six seconds). When distorted by irrelevant speech, short queries become very challenging. In fact, speech added to these queries makes the original signal hardly perceptible to humans.

In order to study the influence of the number of successive closest frames (SCF) on the system performance, Table 2.3 shows the min NDCR generated with Global Mean using different SCF values (SCF-# denote the number of successive closest frames used in the search algorithm). From this table we notice that the worst result is achieved with SCF-0 (i.e. we don't take into consideration any frame before and after the closest frame). The best result is given by SCF-1, which reduces the average min NDCR for all transformations from 0.214 to 0.181.

On the other hand, we notice that the min NDCR increases when we use two or more neighboring frames (SCF-2 and SCF-3). This can be explained by the fact that the count is updated not only for the real nearest query frame but also for the neighboring frames. For example, if the real nearest query frame to the reference frame is query frame 3, and the

search algorithm found that the nearest query frame is frame 4, then the count is updated not only for frame 4 but also for frames 2, 3, 5 and 6 (with SCF-2). Thus, as we increase the SCF value, we increase the likelihood of finding the real nearest neighbor, but we also increase the count for erroneous matches.

Table 2.3 Min NDCR generated with Global Mean fingerprint using different SCF values

SCF	T1	T2	T3	T4	T5	T6	T7	Average
0	**0.075**	0.142	0.201	0.149	**0.201**	0.425	0.306	0.214
1	**0.075**	**0.075**	**0.179**	**0.127**	**0.201**	0.343	0.269	**0.181**
2	0.082	0.097	**0.179**	0.134	0.216	0.366	**0.246**	0.188
3	0.09	0.097	**0.179**	0.134	0.209	**0.336**	0.284	0.189

To evaluate the effectiveness of converting the spectrogram into binary images, Table 2.4 shows the min NDCR obtained using the best parameters (SCF-1 and 48 dimensions) with and without converting the spectrogram matrix into binary images. In the case where the spectrogram matrix is not converted into a binary image, each $w \times h$ frame taken from the spectrogram matrix is directly converted into a d-dimensional vector. Thus, each element of this vector represents the sum of the intensity values contained in the horizontal or vertical slices of the spectrogram matrix (instead of the binary image). In other words, instead of converting the $w \times h$ binary image into a d-dimensional vector, we convert the $w \times h$ spectrogram matrix (represented in red rectangle in Figure 2.2) into a d-dimensional vector.

Table 2.4 Min NDCR per transformation obtained with Global Mean feature with/without using binary images and with only one fingerprint version instead of four versions

	T1	T2	T3	T4	T5	T6	T7	Average
Without binary images	0.104	0.097	0.784	0.806	0.97	0.948	0.955	0.666
With binary image	0.082	0.075	0.403	0.254	0.194	0.545	0.41	0.280

It can be seen from Table 2.4 that the average min NDCR over all transformations goes down from 0.666 (no quantization) to 0.280 with quantization into a binary image. The degradation

in min NDCR without quantization is not significant for audio transforms T1 and T2. This is not the case for the other transformations, especially for the ones that add irrelevant speech to the queries. The reason is that any energy changes of the spectrogram matrix will be encoded into the fingerprint for the unquantized version making it sensitive to any noises added to the signal. Consequently, the *distance* between two fingerprints will be affected depending on the difference in the energy values.

On the other hand, converting the spectrogram matrix into binary images discards the real energy values of the signal and reduces the impact of the energy value changes. Even when a noise is added to the signal, binary quantization strategy prevents the noise from being encoded into the fingerprint when the spectral value does not exceed the threshold. Even when the noise forces the spectral value to exceed the threshold, its real intensity value is replaced by 1 in the binary image (regardless of the real intensity value), which reduces the possibility of obtaining a large *distance* between the transformed fingerprint and the original fingerprint.

The last line of Table 2.4 shows that the average min NDCR obtained with SCF-1 and 48 dimensions when using only one version of the binary image is 0.280, which is significantly higher than 0.181 (see Table 2.3, SCF-1) obtained with four versions of the Global Mean. This shows that our strategy of using four different Global Mean fingerprints derived from four different thresholds works very well.

Since we have optimized our system on TRECVID 2010 dataset, we used the TRECVID 2009 dataset to validate the performance of our system using the Global Mean fingerprints. Table 2.5 shows the min NDCR, the number of missed queries and Mean F1 achieved on TRECVID 2009 dataset using Global Mean fingerprint with 48 dimensions and SCF-1.

Table 2.5 Min NDCR, number of missed queries and Mean F1 for Global Mean fingerprint on TRECVID 2009

	T1	T2	T3	T4	T5	T6	T7	Total/Average
Min NDCR	0.067	0.082	0.127	0.09	0.119	0.246	0.187	0.131
Missed queries	9	10	16	11	14	23	23	106
Mean F1	0.857	0.861	0.833	0.857	0.833	0.845	0.828	0.844

From Table 2.5, we can see that on TRECVID 2009 Global Mean based fingerprints result in a min NDCR averaged over all transformations of 0.131, which is better than the average min NDCR of 0.181 achieved on TRECVID 2010 dataset (see Table 2.3). From a total of 1407 queries, our system missed only 106 queries and correctly detected 1301 queries (more than 92% correct detection).

The worst results in terms of min NDCR and missed queries are achieved with difficult transforms T6 and T7 that add irrelevant speech to the queries. Nevertheless, good results are obtained for transform T5, which also adds irrelevant speech to the queries. The proposed system achieved a good localization accuracy averaged over all transformation of 0.844 (1.0 is the best possible localization accuracy). Note that the Mean F1 (localization accuracy) did not degrade from transforms T1 to T7 and resulted in a similar performance for all the transformations.

2.6.2 Results for Local Mean Fingerprint

Like Global Mean, we used a 1-sec window frame with 24 ms frame advance with the Local Mean feature, and we also generated four different fingerprint versions. To convert the 1-sec window into binary image, we applied a tile of size $w_b = 16 \times h_b = 12$ for a total of 462 tiles (see Figure 2.4 for w_b and h_b definition).

In the search step with the Global Mean fingerprints, we compared each reference version to all query versions. However, for the Local Mean fingerprints, we compare each reference

version to the corresponding query version generated with the same threshold only. This is because we have noticed that, unlike Global Mean, Local Mean provides poor score when we compare reference fingerprints to query fingerprints generated using different thresholds. The thresholds used to create these versions are: *mean* of the local spectrogram matrix segment (i.e. 12 × 16 local matrix segments), 0.6 × *local mean,* 0.4 × *local mean* and 0.2 × *local mean*.

Table 2.6 shows the min NDCR for Local Mean fingerprint generated with different thresholds using two SCF values: SCF-1 and SCF-3.

Table 2.6 Min NDCR for Local Mean fingerprint for different thresholds and different SCF values when tested on TRECVID 2010 dataset

	Threshold	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**Average**
SCF-1	**mean x 0.2**	0.194	0.231	0.254	0.201	0.343	0.373	0.351	0.278
	mean x 0.4	0.201	0.216	0.216	0.201	0.328	0.336	0.313	0.259
	mean x 0.6	0.172	0.201	0.216	0.179	0.612	0.313	0.619	0.330
	mean	0.142	0.179	0.194	0.164	0.276	0.306	0.642	0.272
	combined	0.149	0.179	0.209	0.157	0.284	0.313	0.276	0.224
SCF-3	**mean x 0.2**	0.194	0.246	0.231	0.201	0.336	0.373	0.358	0.277
	mean x 0.4	0.194	0.216	0.216	0.187	0.313	0.358	0.321	0.258
	mean x 0.6	0.187	0.194	0.209	0.187	0.313	0.321	0.284	0.242
	mean	0.724	0.179	0.194	0.179	0.276	0.299	0.5	0.336
	combined	0.149	0.187	0.201	0.164	0.269	0.313	0.478	0.252

As we can see from Table 2.6, for most transforms, the min NDCR reduces as we increase the local mean based threshold. Fingerprints generated with high threshold contain less information, but result in higher matches between query and reference fingerprints. In fact, the lowest min NDCR is achieved with the threshold equal to the local *mean* for both SCF-1 and SCF-3 feature parameters for many transforms. However, for some transformations, when threshold is set to the local mean it decreases the performance. This is the case with T1

(min NDCR = 0.724) and T7 (min NDCR = 0.642) where the min NDCR is very high compared to the rest of the local mean fingerprint versions. The reason is that some queries with music contain repeated music segments (i.e. the same music segment repeated within the same audio at different times). However, the ground-truth contains the start time and finish time of only one of these music segments. In other words, to be considered as a true positive, the segment found by the copy detection system should overlap with this ground-truth (same start and finish times). An example of this problem is illustrated in Figure 2.10 where the audio signal contains two identical segments (*a* and *b*), but only one of them matches the ground-truth. For some of these queries, our system detects the correct audio segment with a high score, but the start and finish times do not overlap the segment in the ground-truth. In such a situation, the decision threshold that rejects false alarms becomes very high resulting in a much higher min NDCR.

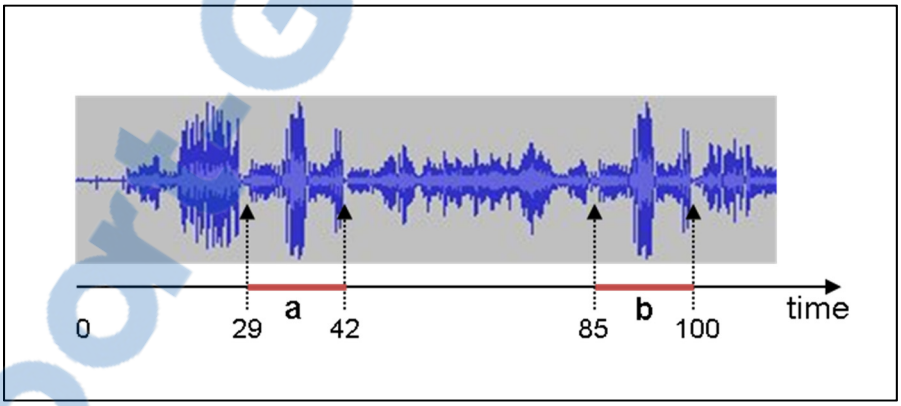

Figure 2.10 Example of the repeated segment problem

By using the best of the four local mean fingerprint versions (combined results represented in grey rows in Table 2.6), we reduce the min NDCR averaged over all transformations. In the same time, the problem of repeated music segments is solved.

The lowest averaged min NDCR for all transformations achieved by Local Mean fingerprint is equal to 0.224 compared to 0.181 achieved by the Global Mean fingerprint, which is 19% lower than Local Mean. In fact, min NDCR for Global Mean is significantly lower than for

Local Mean for all transformations except T6. The poor performance of Local Mean relative to Global Mean was surprising since our preliminary experiments on queries missed by Global Mean showed good results. When we examined our result in-depth, we noticed that although Local Mean detected many queries missed by Global Mean, it also missed other queries detected by Global Mean. In fact, many of the queries missed by Local Mean are audio transformations where different parts of the audio signal have been replaced by silences. It seems that Global Mean is invariant to such a transformation. In the NN-based system (Gupta, Boulianne and Cardinal, 2012), the silent segments are located using a voice activity detector, and then skipped when computing the matching counts.

2.6.3 Combined Results from Global Mean and Local Mean Fingerprints

In order to lower the min NDCR, we combined the results of Global Mean and Local Mean features (we used all versions for both features). We combined the results by first generating separately the best results for each fingerprint, and then keeping the results with the highest matching counts (matching counts as shown in Figure 2.9). Table 2.7 shows the results of this combination using SCF-1 and SCF-3 parameters.

Table 2.7 Min NDCR for combined features on TRECVID 2010 dataset

	T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**Average**
LM1.GM1	**0.075**	**0.075**	0.127	**0.09**	0.194	0.291	**0.239**	**0.156**
LM1.GM3	0.09	0.097	0.149	0.112	0.209	0.291	0.276	0.175
LM3.GM1	**0.075**	**0.075**	**0.112**	0.097	**0.187**	**0.261**	0.396	0.172
LM3.GM3	0.09	0.097	0.142	0.097	0.201	0.276	0.261	0.166

The lowest average min NDCR over all transformations is obtained with *LM1.GM1* (combination of Local Mean and Global Mean features using SCF-1 value). LM3.GM1 also gives good results and achieved the lowest min NDCR for five transformations. Notice that these two configurations use Global Mean with the SCF-1 parameter that gave the best results for a single feature (see Table 2.3 for Global Mean feature).

As mentioned above, the results for combined features use all the versions of each fingerprint. We conducted another test using all versions of Global Mean and only one version of Local Mean (this version is generated with a threshold of 0.6 × local mean). We used SCF-1 for Global Mean feature and SCF-3 for Local Mean feature. This new configuration, denoted as *LM3.GM1**, reduces the average min NDCR from 0.156 (LM1.GM1) to 0.147.

Figure 2.11 compares the min NDCR obtained with LM3.GM1* to the best results obtained for each fingerprint separately. It can be seen from this figure that LM3.GM1* achieved the lowest min NDCR for all transformations when compared with Global Mean and Local Mean features separately. The average min NDCR over all transformation is reduced by 18% compared to Global Mean and 34% compared to Local Mean.

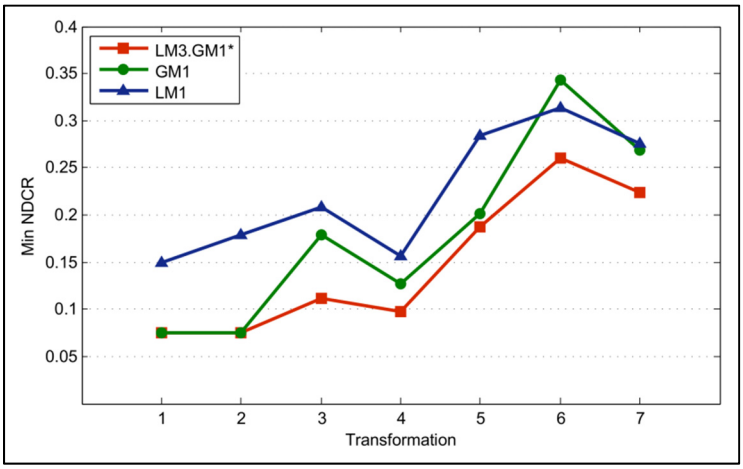

Figure 2.11 Comparison of the best results for each feature separately and for their combination

2.6.4 Results for Salient-Regions Fingerprint

In order to reduce the run time, we generated only one version using the proposed Salient-Regions fingerprint with a threshold equal to the spectral mean, compared to four versions generated with Global-Mean and Local-Mean based features.

Table 2.8 shows min NDCR for Salient-Regions fingerprint averaged over all the seven audio transformations using varying SCF values when tested on TRECVID 2010 dataset. As expected, the min NDCR decreases with increasing dimension. However, unlike our previous experiments with Global Mean and Local Mean, the best results are achieved with SCF-0. The new feature seems to better represent the binary image and generates different fingerprints for successive frames, reducing the impact of *similar successive frame* problem.

Table 2.8 Min NDCR for Salient-Regions fingerprint averaged over all transformations with varying SCF values

Dimension	**SCF-0**	**SCF-1**	**SCF-3**
12	0.149	0.145	0.149
24	0.135	0.136	0.143
44	0.129	0.129	0.132

In order to study the impact of converting the spectrogram matrix into binary images on Salient-Regions performance, we performed experiment similar to that for the Global Mean fingerprints. Figure 2.12 compares the min NDCR for each transformation using Salient-Regions fingerprint (with 24 and 44 dimensions) when we convert the spectrogram matrix into features with or without the binary image transformation step. Note that without binary images, salient regions are selected directly from the spectrogram matrix based on their real energy values. Binary image generation technique is useful in reducing signal noise and allows the generation of multiple fingerprint versions of the same audio file (like Global Mean and Local Mean fingerprints).

From Figure 2.12 we see that the impact of binary images for Salient-Regions performance is similar to the Global Mean one. Extracting salient regions from binary images reduces the min NDCR significantly for transformations that add irrelevant speech to the query (T5, T6 and T7), regardless of the number of dimensions. However, this strategy gives slightly higher min NDCR for transformations that do not add irrelevant speech to the query (except for T4).

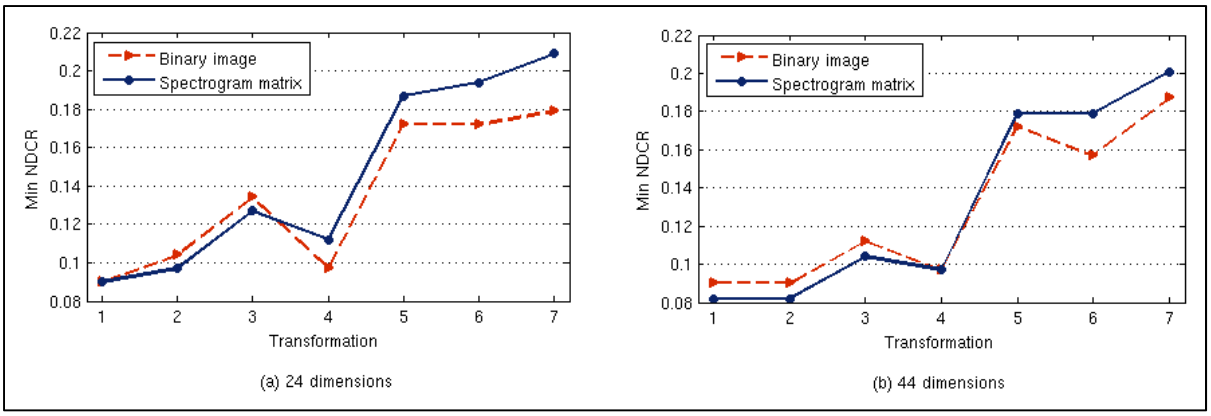

Figure 2.12 Min NDCR for Salient-Regions when extracting features from binary images or spectrogram matrix, for (a) 24 dimensions and (b) 44 dimensions

Table 2.9 compares the best results given by Salient-Regions (SR), Global Mean (GM) and Local Mean (LM) features when tested on TRECVID 2010 dataset. From this table it can be seen that Salient-Regions fingerprints with 44 dimensions (SR-44) outperforms both fingerprints and decreases the min NDCR (averaged over all transformations) by 29% compared to Global Mean fingerprint and by 42% compared to the Local Mean fingerprint.

Table 2.9 Min NDCR for Salient-Regions (SR) fingerprint with varying dimensions compared to Global Mean (GM) and Local Mean (LM) fingerprints

Feature	T1	T2	T3	T4	T5	T6	T7	Average
SR-12	0.104	0.112	0.149	0.112	0.179	0.187	0.201	0.149
SR-24	0.09	0.104	0.134	**0.097**	**0.172**	0.172	**0.179**	0.135
SR-44	0.09	0.09	**0.112**	**0.097**	**0.172**	**0.157**	0.187	**0.129**
GM	**0.075**	**0.075**	0.179	0.127	0.201	0.343	0.269	0.181
LM	0.149	0.179	0.209	0.157	0.284	0.313	0.276	0.224

Although the lowest min NDCR averaged over all transformations is given by SR-44, SR-12 and SR-24 also give good results and lower the min NDCR compared to Local Mean and Global Mean. However, the lowest min NDCR for T1 and T2 transformations is achieved with the Global Mean feature. Global Mean gives good results for transformations that do not add irrelevant speech to the query. However, its performance degrades for transformations

that add irrelevant speech. Salient-Regions feature reduces the impact of adding irrelevant speech to the queries and achieves results comparable to those transformations that do not add irrelevant speech.

2.6.5 Comparative Audio Copy Detection Systems

Figure 2.13 compares the performances of the best results achieved by our method (SR-44) with the NN-based (Gupta, Boulianne and Cardinal, 2012) and Shazam (Wang, 2003) systems, when tested on TRECVID 2010 dataset. It can be seen from Figure 2.13(a) that SR-44 achieved the lowest min NDCR followed by NN-based system. In fact, the min NDCR averaged over all transformations goes down from 0.193 for NN-based system to 0.129 for SR-44.

On the other hand, Shazam gave the worst results with a min NDCR averaged over all transformations equal to 0.488. When we look at Shazam results, we found that a large number of missed queries are queries of type (2) (reference video embedded in a non-reference video, see Section 1.2.1 for more detail). However, Shazam achieved better localization accuracy for the detected queries than NN-based as shown in Figure 2.13(b).

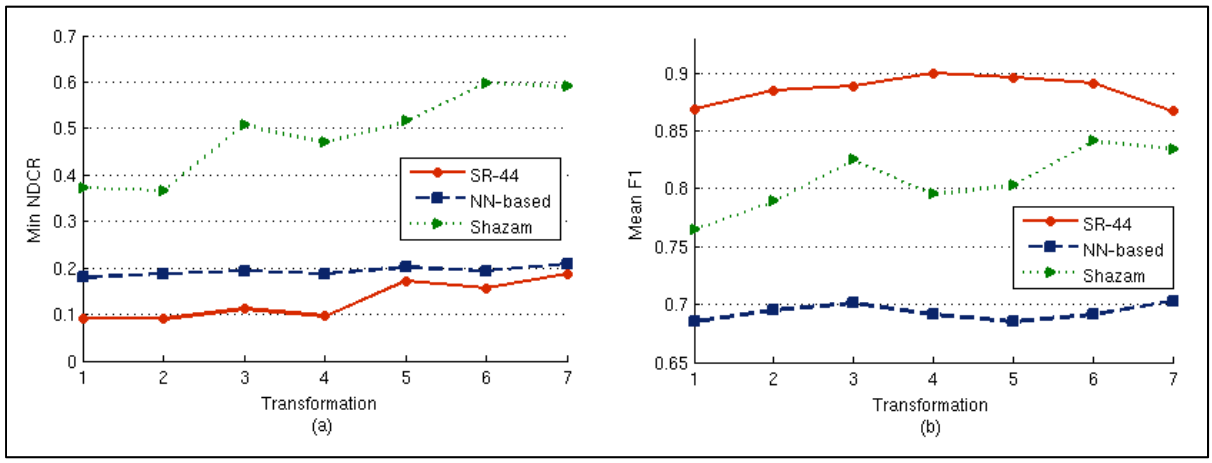

Figure 2.13 Salient-Regions, NN-based and Shazam evaluation on TRECVID 2010 dataset: (a) min NDCR and (b) Mean F1

As for mean F1, our method gave the best localization accuracy with Mean F1 averaged over all transformations of 0.885 compared to 0.807 and 0.693 for Shazam and NN-based systems respectively.

Running time in seconds (per query averaged over all transformations) versus min NDCR (averaged over all transformations) for SR-12, SR-24, SR-44, NN-based and Shazam systems on TRECVID 2010 dataset are shown in Figure 2.14. This figure shows that Shazam is the fastest system and requires less than 2 seconds, on average, to process a query. NN-based system takes roughly 170 secs/query compared to 50 secs/query for SR-12, which is 3.5 times faster. SR-24 also gave a reasonable run time of 95 secs/query, while SR-44 used the highest run time of 440 secs/query.

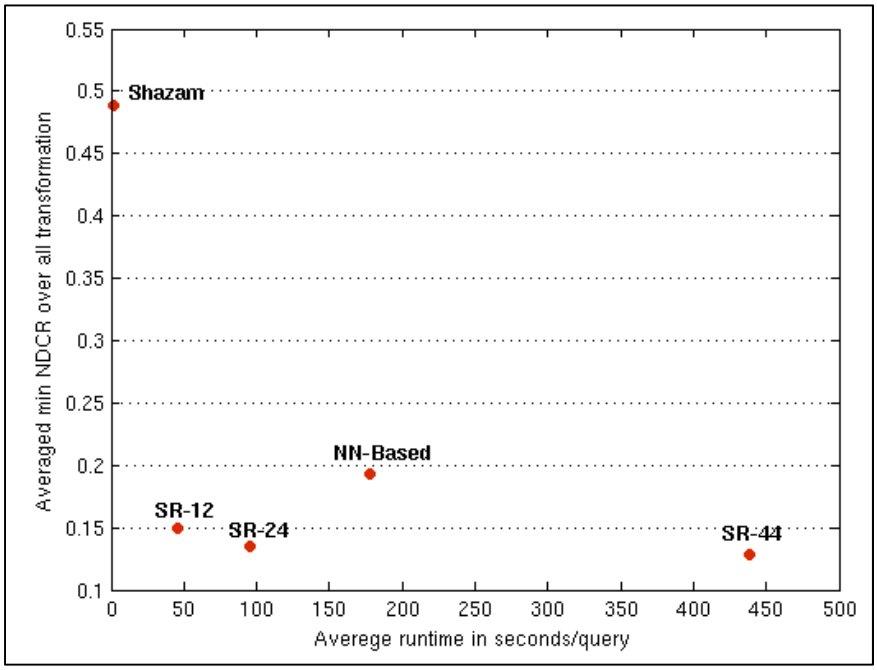

Figure 2.14 Average run time (in secs/query) versus min NDCR (averaged over all transformations) for different systems on TRECVID 2010 dataset

The run time for Salient-Regions based system increases with the number of dimensions, but not proportionately. For example, run time is multiplied by a factor of 10 when using SR-44 instead of SR-12, even though the number of dimensions is only four times greater. This

anomaly is related to the software implementation of the similarity search algorithm on the GPU, and is primarily due to memory limitations on the GPU that leads to this drastic increase in run time. In Chapter 4 we will describe the problems related to the GPU memory limitations, and we will provide several solutions to optimize the use of GPU memories.

Figure 2.15 shows the number of missed queries per audio transformation for SR-44, NN-based and Shazam systems when tested on TRECVID 2009 dataset. We notice that Shazam system missed 845 queries compared to 76 and 58 missed queries by SR-44 and NN-based systems respectively. In fact, Shazam returns the correct reference files for many queries, but with wrong localizations (considered as false positive in TRECVID). To reduce the impact of this situation, we tested Shazam without localization of the query in the reference file (i.e. true positive if the returned true reference file is ranked first regardless of query location). We refer to this scenario as *Shazam**.

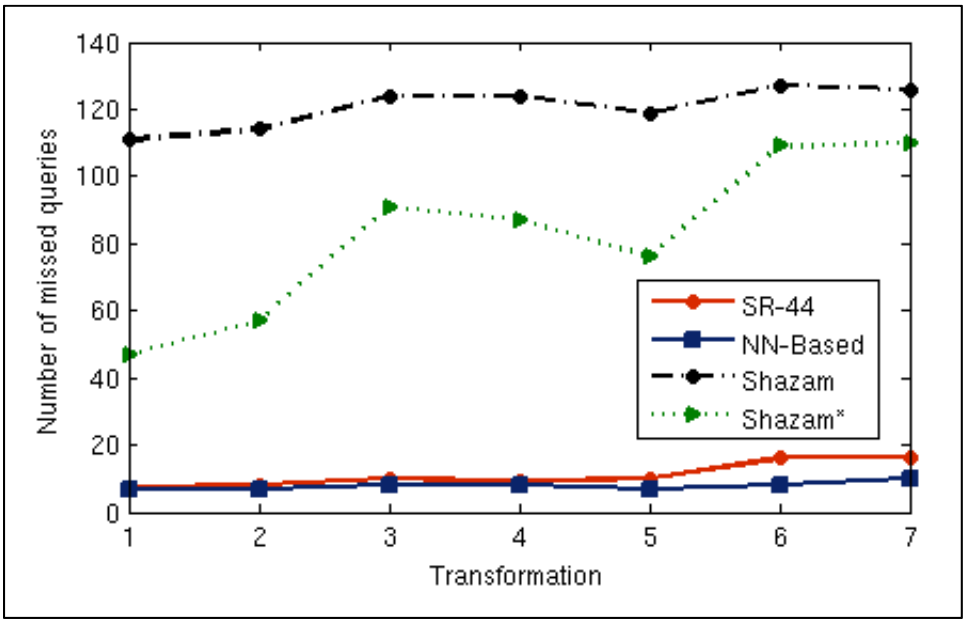

Figure 2.15 Number of missed queries per transformation for different systems on TRECVID 2009 dataset

Figure 2.15 shows that Shazam* detects overall 268 more queries compared to Shazam. Even with this improvement, Shazam system missed significantly more queries than SR-44 and NN-based systems.

The min NDCR averaged over all transformations is equal to 0.06 for NN-based system compared to 0.09 for SR-44 when evaluated on TRECVID 2009 dataset. The principal difference between NN-based and SR-44 is for transformations that add irrelevant speech to the queries. For these transformations, a number of queries detected by NN-based while missed by SR-44 are short queries. For these queries, the resulting number of matching frames between the query and the reference is very low for both the systems. However, NN-based system successfully returned the true reference file corresponding to the short query in first position while the SR-44 did not. In these cases, SR-44 gave higher scores to an imposter reference and therefore failed to return the correct query in the first position for these short queries.

We notice that NN-based and SR-44 systems give lower min NDCR on TRECVID 2009 dataset than on TRECVID 2010 dataset. One of the reasons is the presence of queries that are speeded up or slowed down[3] (compared to the reference query) in TRECVID 2010 dataset. For this reason we have generated three fingerprint versions (original, speeded up and slowed down fingerprint versions, see Section 2.4) to reduce the influence of these time-frequency scale changes. NN-based system detected a number of these queries without using the speeded up or slowed down versions of the fingerprint. In fact, the NN-based search algorithm finds small segments of the query that match the reference (where the number of matching frames is above a certain threshold) and combines them for the final results. Our search algorithm, in contrary, finds only one segment that has the highest number of matching frames.

[3] Time-frequency scale modification as explained in Section 2.4.

In addition, many TRECVID 2010 queries have parts of the original query replaced by small silent segments at different places in the query. This complicates the task, especially when the query is combined with other transformations such as “mixed with speech”. NN-based system overcame this problem by detecting and then discarding all silent segments, whereas SR-based systems seem to be robust to such transformations.

2.7 Summary

In this chapter we have described our audio fingerprinting system that uses fingerprints derived from a spectrogram matrix. We have introduced Global Mean, Local Mean and Salient-Regions fingerprints, which are generated from binary images. These binary images are obtained by converting the audio signal into spectrogram matrix, and then converted to binary images by using a threshold. We have shown that using different fingerprint versions of Global Mean and Local Mean result in significantly better performance. In fact, multiple versions of fingerprints suppress noise to a varying degree. This varying degree of noise suppression improves the likelihood of one of the images matching a reference image. However, using multiple fingerprint versions increases considerably the run time since every reference version should be compared to each query version. We have shown that using only one Salient-Regions fingerprint version results in better detection performance, while decreasing the processing run time. Each Salient-Regions fingerprint encodes the positions of salient regions of a quantized binary image obtained based on the average of the spectral values. We have shown that this fingerprint is a better representation of the binary image and results in significantly better performance compared to the two other fingerprints.

Second, we have described the search approach we have adopted from the search algorithm used in the NN-based system to retrieve audio fingerprints. This algorithm labels each reference fingerprint with the closest query fingerprint, and then counts the number of matching frames for each alignment between the query and the reference. We have improved this search for Global Mean and Local Mean fingerprint retrieval by associating the nearest

query frame and the *N* nearest successive neighboring frames of the query when counting the number of matching frames.

In addition, we have compared the proposed system using Salient-Regions fingerprint to NN-based and Shazam audio copy detection systems using TRECVID 2009 and 2010 datasets. We have demonstrated experimentally that the proposed system outperforms both systems for all transformations on TRECVID 2010 dataset in terms of min NDCR and Mean F1. On TRECVID 2009 our system achieved a min NDCR of around 0.09, which is comparable to the 0.06 achieved by NN-based system. Note that NN-based system achieved the best results for all transformations for the audio copy detection task in the TRECVID 2009 evaluation campaign.

CHAPTER 3

VIDEO FINGERPRINTING

In this chapter, we introduce a novel video fingerprinting system robust to several video transformations. Video feature extraction is based on the insight gained from the Salient-Regions audio fingerprint generation scheme introduced in the previous chapter. We propose two visual feature extraction methods. We compare them to three other visual features used by two state-of-the-art video copy detection systems. Results of this comparison on TRECVID 2009 and 2010 datasets show the robustness of the proposed visual features, especially for queries that do not include geometric transformations.

3.1 System Overview

The overview of the proposed video fingerprinting system is illustrated in Figure 3.1. First, we convert each reference video into a sequence of images (or frames). Then, we convert each image to greyscale image and we change its size to a fixed size (width = height = 300 pixels). After preprocessing, we extract fingerprints from these images and we store them into a video fingerprints database.

A video query is processed in a similar way, with some additional components. Video queries go through many complex transformations (decrease in quality, picture in picture, insertion of pattern, change of content, etc.), and a video query may contain a combination of these transformations. To cope with these transformations, we introduced several preprocessing components to generate additional query fingerprint versions.

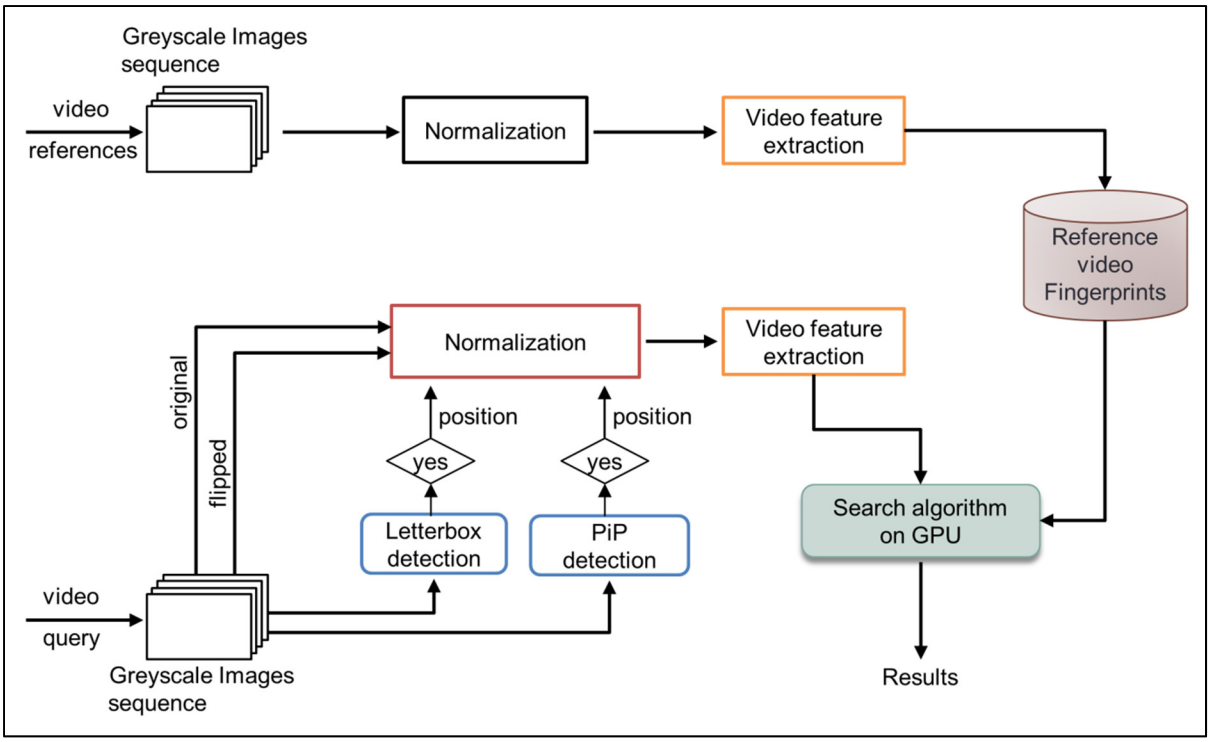

Figure 3.1 Proposed video copy detection system overview

We start by generating fingerprints for the original video query and a flipped version of the video query. The *letterbox detection* component detects and removes (if it exists) black borders from the video.

The Picture in Picture (PiP) transformation is a challenging video transformation. We introduce the *PiP detection* component that detects PiP from the original video query and extracts (if it exists) the foreground video before extracting the features. We describe in the following the feature extraction and these two components (*letterbox and PiP detection*) in detail.

3.1.1 Letterbox Detection

Adding black borders in video query frames is a common video transformation. This transformation, as such, is not challenging since the content of the original video is not

affected. Some approaches (especially those based on local features) are robust to this transformation and do not need to detect letterbox in order to extract fingerprints.

In our system, removing letterbox is a necessary preprocessing step since video fingerprints encode the positions of salient regions. Thus, even if the salient regions of a query image are the same as the reference, their positions will differ (scaled by the letterbox size).

Some works rely on edge information and the temporal intensity variance to detect letterbox (Liu, Liu and Shahraray, 2009; Yeh, Hsu and Lu, 2010). In our system we detect letterbox in a different way. First, we select a fixed number of images from the video query. In our experiments we select 50 images. These images are selected uniformly and regardless of the video length. We process in this way to avoid handling all query frames and limit, therefore, the processing run time. We assume that the letterbox is inserted to all video images, and the letterbox can be detected based on a small number of images. For each selected image, we assign a 0 (black pixel) to each pixel that has pixel intensity below a certain value in order to cope with noise and small intensity value changes. Then, we count the number of black lines (sum of the elements in the line equal to 0) for each border (top, bottom, left and right) and retain the position (for each border) of the last black line before we encounter a non-black line. A letterbox is declared found if the same letterbox positions are detected for at least half of the selected images. When the letterbox is detected successfully, the letterbox detection component provides the position of the detected letterbox region to the *normalization* component in order to remove it from each video image, and the remaining video image is resized to 300 × 300 pixels.

3.1.2 PiP Detection

Handling PiP transformation is a difficult task for most of video fingerprinting systems, even for those that use local features such as the Scale-Invariant Feature Transform (SIFT) descriptors. Although these features are robust to scale change, many approaches have found that it is better to treat PiP transformation separately by detecting PiP (Liu, Liu and

Shahraray, 2009) or by using additional database of half-sized videos in order to increase likelihood of matching the points of interest (Douze, Jegou and Schmid, 2010). A number of approaches are based on the Hough transform (Duda and Hart, 1972) to detect lines as candidates for PiP region boundaries (Bilal Orhan et al., 2008; Liu, Liu and Shahraray, 2009; Mou et al., 2013). We propose a different use of the Hough transform. Our PiP component detects intersection points of perpendicular lines to represent corners of the candidate PiP region rather than lines.

Like the letterbox detection, only a few images are selected to detect PiP (50 images for each video query). We divide each selected image into five regions (four corners and the center), where the size of each region is equal to half of the original image size. We process as follows since a PiP in TRECVID dataset may appear in these locations with different sizes: [0.3, 0.5] of the original size. In addition, this strategy decreases likelihood of detecting perpendicular lines of non-PiP regions and increases, therefore, the number of apparitions of candidate PiP corners. Then, each selected image region is processed as described in Figure 3.2.

1. For each region of the selected images:
    2. Perform edge detection
    3. Perform Hough transform and extract line segments (only horizontal and vertical lines)
    4. Detect locations (pixel positions) of perpendicular lines
    5. Increment the number of the detected locations
6. Merge closest locations
7. Get the top-4 locations (number of apparitions)
8. Verify if these locations form a rectangle

Figure 3.2 PiP detection steps

Figure 3.3 shows the result of performing steps 2-4 of Figure 3.2 on six images (*central* region of the original image), where the intersection points of the detected segment lines are marked with red points. Notice that we keep only horizontal and vertical line segments, and

we extend the extremities of each segment to force the intersection of short segments. For every processed image, we keep the locations of the intersection points and increment the number (score) of their appearances (step 4-5). Once all the images have been processed, we merge the locations and scores of points that are too close (step 6), and keep the intersection points of four points that appear most frequently (step 7). These *top-4* points represent corners of the candidate PiP region. Finally, in step 8, we verify if these corners form a rectangle based on some criteria (size of sides, parallel and vertical sides, etc.).

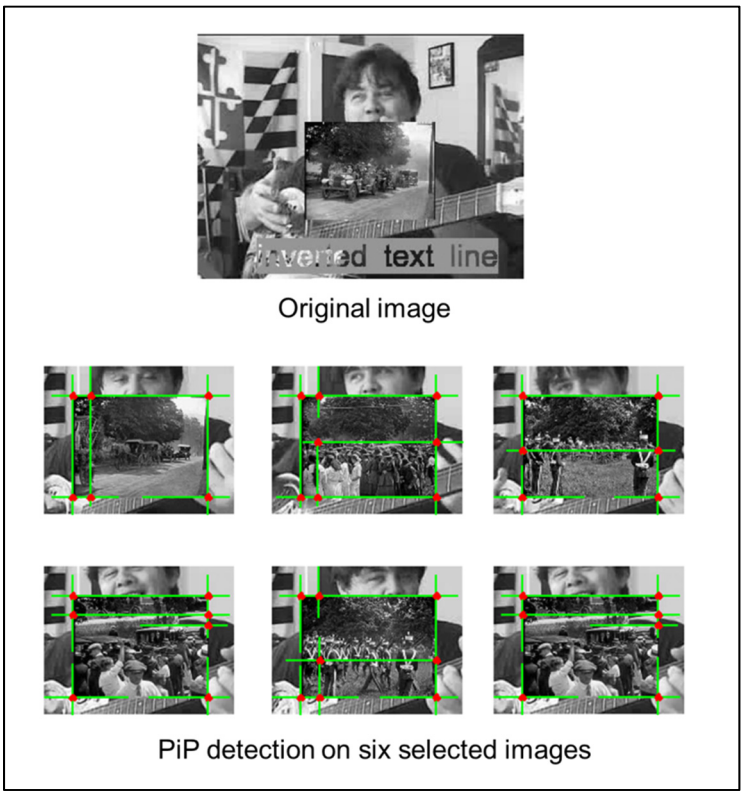

Figure 3.3 Example of PiP detection

3.1.3 Video Fingerprint Extraction

Video feature extraction is based on the same idea as the audio feature extraction: positions of salient regions of an image have good chance to survive signal degradation. The question is how can we define a visual salient region?

Audio fingerprints are extracted from binary images derived from the spectrogram where a value of 1 denotes a time-frequency peak. In other words, a value of 1 indicates the presence of information and a value of 0 denotes its absence. Thus, a salient region is the part of the binary image that has more information than the others. Therefore, multiple audio fingerprints describe the localization of information over time regardless of the real energy values. In contrast, video images are more complex, and each pixel may hold useful information.

We use grayscale images in our system (for robustness against color transformations). Therefore, a pixel value varies from 0 to 255. We propose two different fingerprint extraction schemes: V-intensity and V-motion fingerprints. For these two methods, we apply a square tile of fixed size and compute the sum of the pixel values in each small square of the tile. We divide the image using a tile of 20 × 20 pixels for a total of 225 tiles. Then, each method extracts *d*-dimensional fingerprints as described in the following:

- **V-intensity**: this method relies on the sum of the intensity values of the image squares (here we call them squares instead of tiles since each tile is a 20 × 20 square). However, instead of taking the *top-d* squares like in the case of audio, we look for squares that have average intensity values. To do that, we sort the squares by their values and we take *d/2* squares before and *d/2* squares after the square with the median value. In other words, we take image regions that are neither black nor white, but grey regions. The grey regions are the regions of interest for distinguishing video frames. An illustration of this method is given in Figure 3.4, where the selected regions are represented in grey background. Once these regions are identified, their positions within the image will be used as the final fingerprint.

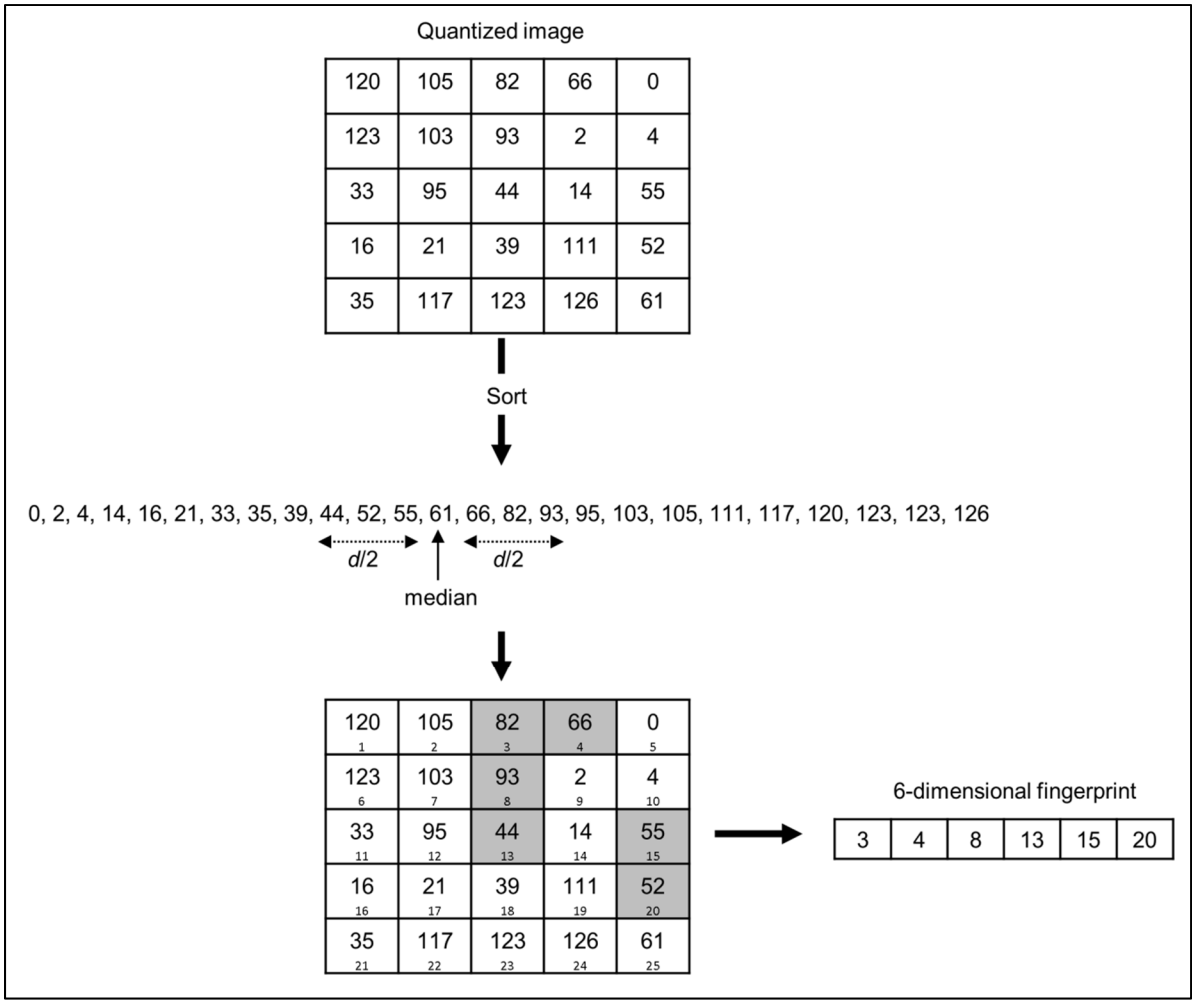

Figure 3.4 Illustration of V-intensity fingerprint generation scheme

- **V-motion**: this method captures a few regions from the image that present the highest difference between two successive images. V-motion fingerprint are detected by looking for the regions that have the highest variations compared to the same regions in the previous frame. Figure 3.5 illustrates the principal steps to extract V-motion fingerprints. In this figure, the difference between *frame 1* and *frame 2* indicates the degree of intensity variations between these two successive frames. The regions that have the highest variations are the salient regions with this method. Finally, like V-intensity fingerprint, V-motion fingerprint encodes the positions of the selected salient regions.

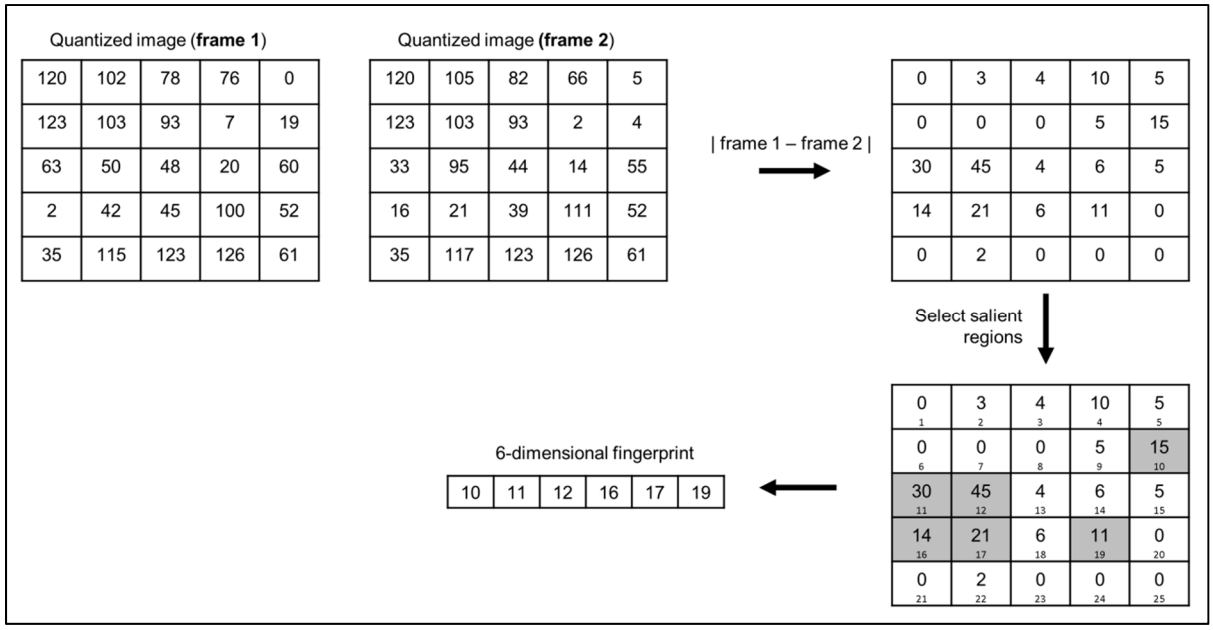

Figure 3.5 Illustration of V-motion fingerprint generation scheme

3.1.4 Matching Algorithm

As stated in the previous subsection, video fingerprints extracted using both the V-intensity and V-motion methods encode the positions of salient regions. This is the same fingerprint representation we have used with audio, where fingerprints encode the position of salient regions derived from the binary images. Thus, the search algorithm used to perform the audio fingerprint retrieval can also be used to retrieve video fingerprints in the same manner.

The only difference between matching audio fingerprints and video fingerprints is the number of tiles obtained when dividing an audio frame or video frame. An audio frame is divided using a tile of size 11 × 11 resulting in 744 regions, whereas a video frame is divided using a tile of size 20 × 20 resulting in 225 regions. In both cases, we select d salient regions, and we compute the similarity between fingerprints of size d.

In Chapter 4, we will introduce two intersection search algorithms that compute this similarity. One of these algorithms needs a vector of size D that corresponds to the total

number of tiles (i.e. 744 in case of audio frame, and 225 in case of video frame) to compute the similarity between two fingerprints. The similarity search is performed on GPU, which is characterized by its memory limitations. Thus, as the value of *D* becomes higher, the search algorithm will slow down since it depends on GPU memory. This fact makes the video fingerprint retrieval faster than the audio fingerprint retrieval, even when we use the same value of *d*. In Chapter 4 we will study in detail the influence of these parameters (*d* and *D*) on the run time of the similarity search algorithm.

Apart from this simple difference, the video fingerprint retrieval works exactly in the same way as the audio fingerprint retrieval described in Section 2.5.

3.2 Results and Analysis

This section evaluates our video copy detection system on TRECVID 2009 and 2010 copy detection datasets. First, we present the results obtained with the proposed system when only video fingerprints are used to detect video copies. We also compare our system to two state-of-the-art video fingerprinting systems that achieved excellent results on TRECVID 2009 and 2010 evaluation campaign. Finally, we give the results achieved by our system on TRECVID 2010 dataset when the audio and video results are combined for the audio+video copy detection task.

3.2.1 Video Only Results

The experimental results achieved using V-intensity and V-motion features are shown in Table 3.1. These results are compared with other systems that used NN-based, DC-SIFT or DCT features.

NN-based system (Gupta et al., 2012) uses the algorithm described in (Gupta, Boulianne and Cardinal, 2012) for the audio copy detection i.e. it maps each frame of the reference to the

nearest frame of the query. However, instead of using MFCCs as audio features, it uses temporally normalized local visual features similar to (Mukai et al., 2010).

In another system, DC-SIFT and DCT are employed conjointly with WASF audio feature for the task of video copy detection (Mou et al., 2013). This system combines the results obtained separately by these features to improve copy detection. We present in this section the results achieved individually by DC-SIFT and DCT for the task of video only detection. In Section 3.2.2, we present the detection performance of this system for the task of audio+video copy detection when the results obtained from all features are combined.

The features presented in Table 3.1 are evaluated on TRECVID 2009 and 2010 datasets using these video transformations: (V1) Simulated camcording, (V2) Picture in picture, (V3) Insertions of pattern, (V4) Strong re-encoding, (V5) Change of gamma, (V6) Decrease in quality, (V8) Post production and (V10) Combination of three different transformations. For more details about these video transformations see Table 1.1.

Table 3.1 Min NDCR by transformation achieved by different visual features on TRECVID 2009 and 2010 datasets

Data	Feature	V1	V2	V3	V4	V5	V6	V8	V10
TRECVID 2009	V-intensity	-	0.351	0.56	0.007	0.007	0.007	0.552	0.448
	V-motion	-	0.284	0	0.045	0	0	0.231	0.313
	NN-Based	-	0.022	0	0.052	0	0	0.037	0.097
	DC-SIFT	-	0.112	0.03	0.09	0.024	0.142	0.201	0.149
	DCT	-	0.224	0.164	0.119	0.104	0.231	0.41	0.306
TRECVID 2010	V-intensity	0.985	0.634	0.276	0.097	0.067	0.149	0.53	0.463
	V-motion	0.896	0.545	0.03	0.082	0.03	0.112	0.321	0.358
	NN-Based	0.6	0.417	0.04	0.18	0.03	0.142	0.187	0.27
	DC-SIFT	0.285	0.154	0.054	0.146	0.038	0.223	0.292	0.2
	DCT	1	0.377	0.246	0.2	0.146	0.323	0.585	0.415

From Table 3.1 we see that V-motion performs better than V-intensity for all transformations and on both datasets (except V4 on TRECVID 2009). Although these two features achieved good results for transformations V3, V4, V5 and V6 that do not include geometric transformations, they give relatively higher min NDCR for transformations that change the content of the images. This is noticeable for transformation V1 (simulated camcording) that gives the highest min NDCR compared to the rest of the transformations. In fact, global visual features are usually sensitive to such types of transformations, as confirmed by the results of DCT feature.

The comparison between DCT and V-motion features shows that V-motion gives better results for most of the transformations on both the datasets. On the other hand, DC-SIFT is better for transformations that modify the content and achieved the best results for V1 (simulated camcording), V2 (Picture in picture) and V10 (Combination of three transformations) transformations on TRECVID 2009 dataset.

Note that NN-based system gives the lowest min NDCR for all transformations on TRECVID 2009 dataset, except for V4 where the best result is obtained by V-intensity. However, on TRECVID 2010 dataset, NN-based system gave the lowest min NDCR for only V5 and V8 transformations. This is primarily due to the videos that contain slide shows (i.e. there is no temporal variability), which results in features that are either zero or one (Gupta et al., 2012). V-intensity suffers from the same problem, in contrast to the V-motion feature that seems to be less sensitive to such videos. In fact, the fingerprints generated from slide shows using V-motion are different, despite the fact that the video images are identical. However, when we examine the pixel values of these video images we found a small difference between some of them. The parts of the image that represent the highest variances become therefore salient regions according to V-motion method.

On TRECVID 2010 dataset, V-motion gave the best results for four transformations that do not change the content of the video (V3, V4, V5 and V6) on both datasets (NN-based system gave the same results for some of these transformations, especially on TRECVID 2009). For

these four transformations, V-motion missed only one query (transformed with V4) on TRECVID 2009 dataset (see Table 3.2). If we compare the number of missed queries to the min NDCR, we see that in some cases different values of min NDCR are obtained for the same number of missed queries. For example, V-intensity and V-motion each missed one query with V4 transformation on TRECVID 2009 dataset, as shown in Table 3.2. However, the min NDCR obtained with V-intensity is 0.007 versus 0.045 with V-motion. Similar behavior is observed with V3 and V6 transformations, where the min NDCR does not reflect the real number of missed queries. This happens when a false alarm has a high score and leads to a high decision threshold (this decision threshold separates true positives from false alarms). Thus, too many queries fall below this decision threshold resulting in a high value for min NDCR.

Table 3.2 Number of missed queries for V-intensity and V-motion on TRECVID 2009 dataset

Feature	**V1**	**V2**	**V3**	**V4**	**V5**	**V6**	**V8**	**V10**
V-Intensity	-	45	14	1	1	0	65	57
V-motion	-	36	0	1	0	0	23	35

In order to evaluate the performance of the technique employed to detect PiP, we count the number of PiPs correctly detected for both datasets (see Table 3.3). As mentioned above in Section 1.2.1, there were 201 PiP queries composed of 134 reference copies and 67 queries from non-reference videos (i.e. to test false alarms). As it can be seen from Table 3.3, the PiP algorithm detects from 73% to 79% of the inserted PiPs.

Table 3.3 PIP detection performance

Dataset	**Detected**	**Missed**	**% of detection**
TRECVID 2009	98	36	73%
TRECVID 2010	106	28	79%

When we looked at the missed PiP copies, we found two principal reasons that prevented the PiP component to detect them. The first problem comes from the *edge detection* step (see Figure 3.2 step 2) that, in some cases, fails to identify the PiP edges as lines by the *Hough transform*. In this work we used *Roberts* method to detect edges. In other words, edge detection could be improved.

The second problem is related to the design of the PiP technique. In fact, to detect PiP that may appear at one of the four corners of the image, we assume that two sides of the image represent sides of PiP candidate. However, we found that many PiPs are not located exactly at the corner of the image, which forms two adjacent rectangles: one represents the PiP and the other is an imposter (see Figure 3.6). In some cases, the intersection points of the imposter rectangle appeared more frequently than the PiP rectangle, leading to missing the PiP. A possible way to solve this problem is to take more than the *top-4* intersection points, and to identify all possible rectangles.

Figure 3.6 Example of PiP detection problem

From the point of view of copy detection, we notice that for TRECVID 2010 dataset, V-intensity and V-motion missed a number of queries where the PiP is correctly detected. For example, V-motion missed 53 queries, even though 25 of them are correctly detected as PiP. When we examined these queries, we found that many of them included *crop* transformation in addition to the PiP transformation (V2). The video Salient-Regions-based fingerprint is

sensitive to such transformations, which increases the min NDCR not only for the missed PiP transformation, but also for V8 and V10 transformations.

On TRECVID 2010 dataset, DC-SIFT achieved the lowest min NDCR for V2 (i.e. PiP) transformation, which is expected since this feature is scale invariant. However on TRECVID 2009 dataset, the best result is achieved by NN-based system that detects most of the V2 copies. NN-based system adopts different strategy for V2 transformation. Instead of detecting PiP, NN-based method extracts 15 additional fingerprints from each video (5 regions × 3 sizes). This technique works very well on TRECVID 2009 dataset, however, the processing run time is increased by a factor of 16.

3.2.2 Audio+Video Results

We use a simple strategy to combine audio and video results. For the audio part, we use the results obtained with the Salient-Regions audio fingerprint generation method introduced in Section 2.3.3. First, we generate the results separately for the audio (we use the audio results obtained with Salient-Regions fingerprint) and video (with V-motion fingerprint) systems. Then for each query, we keep the best result (highest score) achieved by either the audio or the video. In other words, for a given query, if the score of the top reference audio result is higher than the score of the top video reference result, then we take the audio result, otherwise, we take the video result.

The authors of (Gupta et al., 2012) and (Mou et al., 2013) used a more complex fusion strategy, where some priorities are taken into consideration. Audio and video are given equal weight in our fusion technique, and this fusion results in a good performance as shown in Table 3.4 and Table 3.5. In fact, the min NDCR averaged over all transformations is equal to 0.021 for TRECVID 2009 (Table 3.4) and 0.053 for TRECVID 2010 (Table 3.5).

From a total of 9849 queries in TRECVID 2009, our system missed only 122 queries (98.7 % correctly detected). On TRECVID 2010, 347 queries are missed from a total of 11256 queries (96.9 % correctly detected).

Table 3.4 Min NDCR per transformation for audio+video with SR-44 system on TRECVID 2009 dataset

T	Min NDCR	T	Min NDCR	T	Min NDCR	T	Min NDCR
T1	-	**T15**	0	**T29**	0	**T50**	0.022
T2	-	**T16**	0	**T30**	0	**T51**	0.03
T3	-	**T17**	0	**T31**	0	**T52**	0.03
T4	-	**T18**	0	**T32**	0	**T53**	0.022
T5	-	**T19**	0	**T33**	0	**T54**	0.03
T6	-	**T20**	0	**T34**	0	**T55**	0.052
T7	-	**T21**	0	**T35**	0	**T56**	0.045
T8	0.03	**T22**	0.007	**T36**	0	**T64**	0.052
T9	0.03	**T23**	0.015	**T37**	0	**T65**	0.06
T10	0.03	**T24**	0.007	**T38**	0	**T66**	0.075
T11	0.03	**T25**	0.007	**T39**	0	**T67**	0.067
T12	0.03	**T26**	0.007	**T40**	0	**T68**	0.067
T13	0.052	**T27**	0.022	**T41**	0	**T69**	0.09
T14	0.022	**T28**	0.015	**T42**	0	**T70**	0.097

Table 3.5 Min NDCR per transformation for audio+video with SR-44 system on TRECVID 2010 dataset

T	Min NDCR	T	Min NDCR	T	Min NDCR	T	Min NDCR
T1	0.09	**T15**	0.015	**T29**	0.015	**T50**	0.03
T2	0.104	**T16**	0.015	**T30**	0.015	**T51**	0.045
T3	0.112	**T17**	0.015	**T31**	0.015	**T52**	0.037
T4	0.104	**T18**	0.015	**T32**	0.015	**T53**	0.037
T5	0.164	**T19**	0.022	**T33**	0.022	**T54**	0.075
T6	0.157	**T20**	0.022	**T34**	0.022	**T55**	0.067
T7	0.172	**T21**	0.022	**T35**	0.022	**T56**	0.09
T8	0.037	**T22**	0.015	**T36**	0.015	**T64**	0.037
T9	0.045	**T23**	0.015	**T37**	0.015	**T65**	0.045
T10	0.052	**T24**	0.022	**T38**	0.022	**T66**	0.06
T11	0.045	**T25**	0.015	**T39**	0.022	**T67**	0.045
T12	0.097	**T26**	0.067	**T40**	0.06	**T68**	0.104
T13	0.09	**T27**	0.037	**T41**	0.045	**T69**	0.09
T14	0.112	**T28**	0.067	**T42**	0.06	**T70**	0.104

Finally, we compare in Figure 3.7 our audio+video system to the method described in (Mou et al., 2013) (the *Perseus* system) that combines the results obtained by the audio part using WASF feature, and the video results obtained using DC-SIFT and DCT visual features. This method achieved the best results for almost all transformations on TRECVID 2010 campaign compared to the rest of participants (Li et al., 2010).

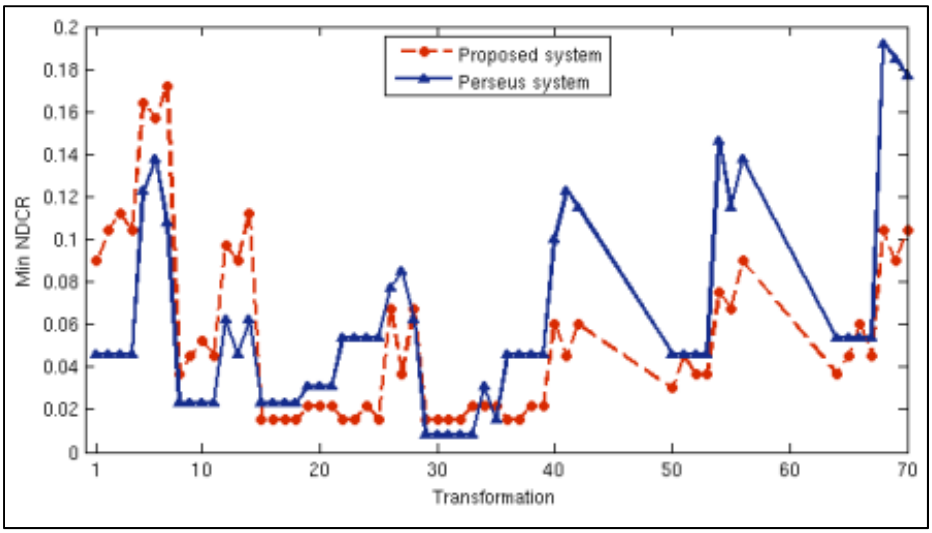

Figure 3.7 Min NDCR of the proposed system for audio+video transformations compared to Perseus system on TRECVID 2010 dataset

It can be seen from Figure 3.7 that our system achieved comparable results to the Perseus system and outperforms it for 35 out of 56 transformations. Furthermore, our system gave better min NDCR averaged over all transformations of 0.056 compared to 0.06 achieved by Perseus system.

3.3 Summary

In this chapter, we have presented a novel video fingerprinting system that can be used to detect video copies subjected to visual transformations. Video fingerprint extraction is similar to the Salient-Regions audio fingerprint extraction introduced in Chapter 2. However, instead of extracting fingerprints from binary images, visual features are extracted from greyscale video images (for robustness to color transformations). We have proposed V-

intensity and V-motion extraction methods and have shown that V-motion is more robust to video transformations than V-intensity. V-intensity selects the salient regions based on the intensity values of the image, while V-motion identifies the regions that represent the highest variation between two successive images. We have compared these two methods to NN-based, DCT and DC-SIFT features using TRECVID 2009 and 2010 datasets. We have shown that V-motion achieved excellent results for all queries that do not include geometric transformations and have outperformed other features for these transformations.

To address the PiP transformation, we have proposed a PiP detection technique that has detected 79% of PiP on TRECVID 2010 dataset. However, a large number of the detected PiPs are missed by the copy detection system. This is primarily due to the additional transformations that change the content of the image as the *crop* transformation. In fact, our method is sensitive to such transformations as the fingerprints encode the positions of salient regions and transformations that change the position of these regions affect the detection performance.

We have also tested our system for the audio+video copy detection task where the queries are transformed by audio and video transformations. These audio+video queries are detected using the best result achieved separately by each system (i.e. the audio and video systems). Then, for each query we have taken the result that has the highest score given either by the audio or the video system. This simple merging technique works very well and gave a min NDCR of 0.021 for TRECVID 2009 and 0.053 for TRECVID 2010.

CHAPTER 4

ACCELERATING THE AUDIO FINGERPRINT SEARCH USING A GPU AND A CLUSTERING-BASED TECHNIQUE

We have described in Chapter 2 and Chapter 3 the Salient-Regions based fingerprint extraction method that encodes the positions of salient regions selected either from binary images for the audio part and from greyscale images for the video part. These fingerprints have shown their robustness against different kinds of audio and video distortions. However, the similarity search is time consuming due to the large amount of data and the high dimensional representation of the fingerprints. Thus, we propose in the present chapter an approach based on the combination of hardware and software techniques that speed up this similarity search by several orders of magnitude. Specifically, we propose GPU implementations of two different similarity search algorithms and we provide a detailed description of an efficient parallel design that reduces compute time by several hundred on a dataset containing over 60 million audio fingerprints. To speed this search even further, we introduce a two-step search based on a clustering technique and a lookup table that reduces the number of comparisons between the query and the reference fingerprints. We also explore the tradeoff between the search execution speed and copy detection performance. We evaluate the performance of the proposed system on TRECVID 2009 and 2010 datasets. Besides, we compare our results in terms of detection performance and search run time to several audio fingerprinting systems.

4.1 GPU Implementation of the Similarity Search

In this section, we first describe the principal concepts of the GPU architecture needed to understand the implementation issues. Then, we present an effective way of using the GPU to label a large set of reference fingerprints with the closest query fingerprint (the similarity search algorithm). We propose two similarity search algorithms suitable for the GPU.

4.1.1 GPU Architecture

A GPU is a graphics device initially designed for graphics processing (video games, graphical user interfaces, etc.). Now GPUs are used jointly with Central Processing Unit (CPU) to accelerate scientific applications. This is greatly facilitated by the Compute Unified Device Architecture (CUDA) created by NVIDIA as a parallel computing platform. Compared to a CPU composed of only a few cores optimized for processing serial tasks, a GPU consists of thousands of cores (smaller but more efficient) designed for handling multiple tasks simultaneously.

CUDA is a heterogeneous programming model where the serial part of the application is executed on the *host* (CPU and its memory), and the parallel part is executed on the *device* (GPU and its memory). CUDA is designed to fully utilize the tremendous capacity of the GPU to process the same function, known as a kernel, on different data. Thus, invoking a kernel creates thousands of threads executing the same kernel code in parallel, referred to as Single Instruction Multiple Threads (SIMT) by NVIDIA.

From design point of view, a thread on the GPU is the smallest execution unit. Threads are grouped into thread blocks. Every thread block can contain up to 1024 concurrent threads. Thread blocks are organized into grids of thread blocks. This thread hierarchy is suitable for the GPU architecture. The GPU launches grids of thread blocks. Each streaming multiprocessor (SMP) executes thread blocks (GTX 580 has 16 SMPs), and cores in the streaming multiprocessor execute threads. In GTX 580 each SMP has 32 cores.

Besides the thread hierarchy, GTX 580 GPU has multiple memory spaces: global memory (1.5 Gigabytes), shared memory (48 Kbytes/SMP shared by all the blocks in the SMP), 32-bit registers (32k registers per SMP) and local memory (512 Kbytes/thread). Global memory is visible to all the threads, including the CPU, but is relatively slow. On the other hand, shared memory is small but very fast compared to global memory. Shared memory is visible to all the threads within a block, which allows them to communicate and share data. The

register memory is very fast, however, each thread can use only a few registers. Finally, each thread can use local memory (slower) when registers are not enough to store local thread variables. An illustration of the thread hierarchy and the memory hierarchy described above is given by Figure 4.1.

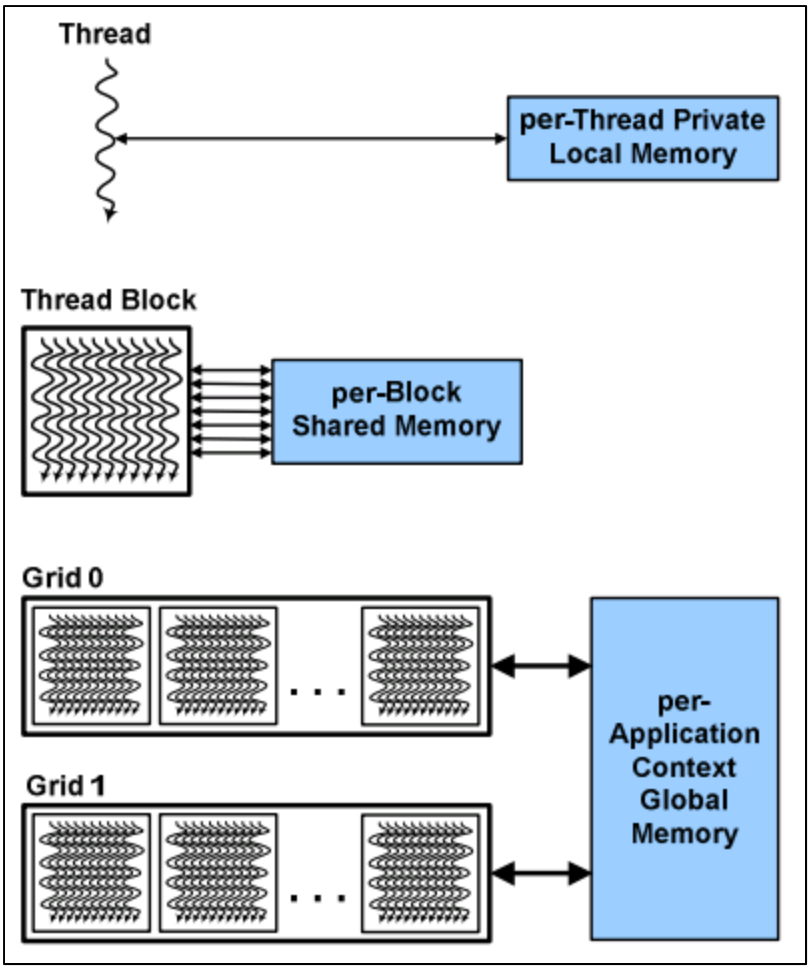

Figure 4.1 CUDA Hierarchy of threads, blocks, and grids, with corresponding per-thread private, per-block shared, and per-application global memory spaces
Taken from (NVIDIA, 2015)

The GPU architecture is more complicated than what is described in this section. We have presented here only the elements necessary to understand this chapter. For example, there exist other types of GPU memory such as the constant memory and the texture memory that can be beneficial for specific types of applications. In our work, we did not use such type of

memory, for this reason we did not describe them in this manuscript. Likewise, different access memory patterns have been designed and should be taken into consideration when using the GPU. Such patterns include coalesced and non-coalesced memory access, and it is important that threads within a warp (groups of 32 threads executed in parallel) access data in coalesced fashion to improve performance.

4.1.2 Similarity Search on GPU

As stated in the previous subsection, CUDA supports heterogeneous computation where the CPU invokes a kernel that performs the parallel portion of the application on the GPU. Since the GPU cannot access data directly from the host memory, input data is transferred from the CPU memory to the GPU memory, and we then copy the results from the GPU to the CPU. Figure 4.2 presents the main steps used in our design to transfer data between the host and the device on CUDA.

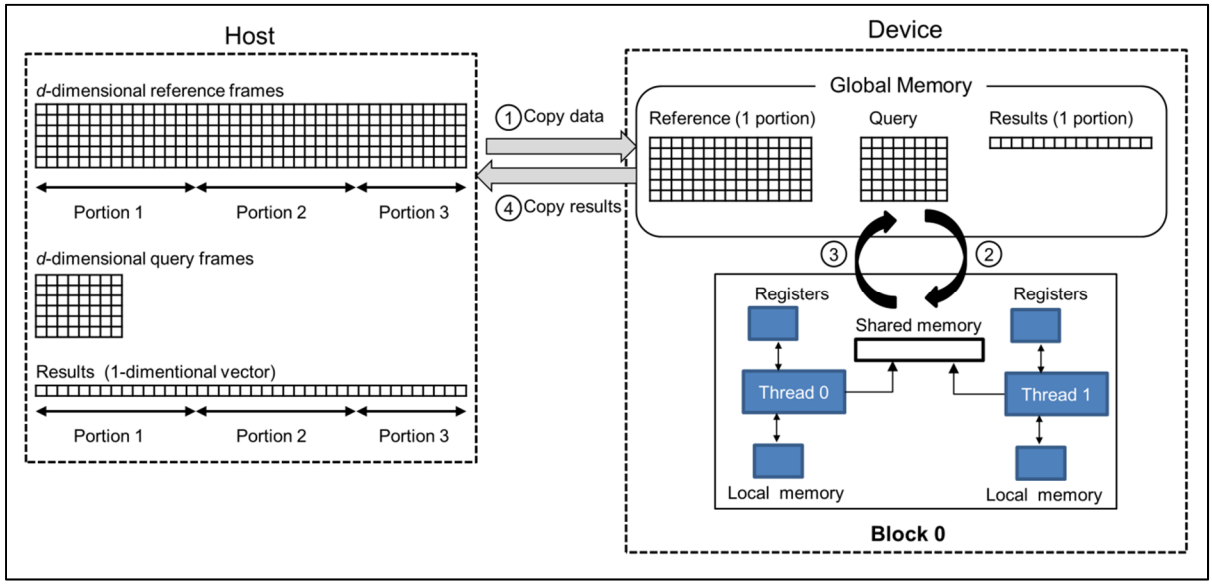

Figure 4.2 Processing flow on CUDA

The similarity search executed on the GPU finds the closest query frame for each reference fingerprint. Therefore, we transfer reference and query fingerprints to the GPU and transfer

back to the CPU a vector of one dimension that contains the closest query frame for each reference fingerprint. We have approximately 61 million reference fingerprints from 400 hours of audio requiring 61,000,000 × d × 4 bytes of memory (d is the number of dimension of the fingerprint). Even for d = 12, there is not enough global memory to transfer all the reference frames. Therefore, we divide the reference frames into smaller portions that can fit into the global memory.

The similarity search as implemented on the GPU is described in Figure 4.2. First, we transfer a portion of the reference fingerprints and all query fingerprints to the global memory. Second, we copy data from global memory to the appropriate GPU memory spaces (shared memory, local memory, etc.) and execute the similarity search on the GPU. Then, we copy the results to the global memory. Finally, we copy back the results to the CPU memory. We repeat these steps until no more reference fingerprints need to be processed.

GPUs are efficient when the problem to be solved is highly parallelizable, which is the case for our fingerprint search algorithm. This search computes the similarity between each reference fingerprint and all the fingerprints corresponding to a query, and it labels the reference fingerprint with the frame number of the closest query fingerprint. To map this similarity search to the GPU architecture, each thread handles one reference fingerprint: the thread fetches the corresponding reference and query fingerprints from the global memory, computes the similarity between the reference fingerprint and all the query fingerprints and finds the closest query fingerprint and returns the frame number of this query fingerprint. The question is where should the thread store the fingerprints in order to minimize the overall compute time?

Global memory is slow, so we should minimize fetching data from global memory when possible. Since each query fingerprint is accessed as many times as the number of reference fingerprints, a possible solution is to copy query fingerprints to the shared memory. Shared memory has the advantage of being fast compared to global memory and is accessible to all the threads in the same block, but has a limited size. A common strategy is to partition data

into subsets that fit into the shared memory. In other words, each thread loads one reference fingerprint to its local memory and one query fingerprint to the shared memory, so each thread within a block can access all the query fingerprints loaded into shared memory. After processing one portion of the query, the threads load and process the next portion until the end of the query. In Section 4.3.1, we will see that this strategy does not compute as fast as keeping all the query fingerprints in the global memory.

4.1.3 Similarity Algorithms

As stated before, the Salient-Regions fingerprints encode the *positions* of salient regions of the binary images. Thus, the similarity between two fingerprints is defined as the intersection between the elements of these two fingerprints. We experimented with two different intersection algorithms on the GPU to compute the similarity given by equation (2.2).

- **Hashing-based algorithm** (see Algorithm 1): this algorithm converts one d-dimensional fingerprint into a vector of D dimensions ($d << D$), and then looks for matching entries in the two D-dimensional vectors. (See Figure 2.7) for definition of d and D). This solution has a linear time complexity. However, memory is a critical commodity on a GPU, and a D-dimensional vector, for every thread, may drain GPU's resources (in our experiments $D = 744$ compared to a maximum $d = 44$).

Algorithm 1 hashing-based

Input: Two vectors $v1$ and $v2$ of size d
Output: The similarity between $v1$ and $v2$

```
1: sim = 0
2: create a vector hash of size D
3: for i = 1 to D do
4:     hash[i] = 0
5: end for
6: for i = 1 to d do
7:     hash[v1[i]] = 1
8: end for
9: for i = 1 to d do
10:    sim = sim + hash[v2[i]]
11: end for
12: return sim
```

- **Sorting-based algorithm** (see Algorithm 2): The *d* elements of the two fingerprints are stored in a sorted order. Then the algorithm iterates through them to compute the number of matching elements. This algorithm takes less memory compared to hashing-based algorithm, but requires more computing.

Algorithm 2 sorting-based

Input: Two sorted vectors $v1$ and $v2$ of size d
Output: The similarity between $v1$ and $v2$

```
1: sim = 0
2: a = 0
3: b = 0
4: while (a < d and b < d) do
5:     if (v1[a] < v2[b]) then
6:         a = a + 1
7:     else if (v1[a] > v2[b]) then
8:         b = b + 1
9:     else
10:        sim = sim + 1
11:        a = a + 1
12:        b = b + 1
13:    end if
14: end while
15: return sim
```

The main processing steps of the similarity search on GPU are described in Figure 4.3**.** In this figure, each thread loads one query fingerprint for a total of *k* query fingerprints per block

(Step 2). In our experiments, we study the impact of loading only one query fingerprint to the shared memory instead of k fingerprints. In addition, this simple modification allows us to test the impact of hashing a query fingerprint into a D-dimensional vector instead of a reference fingerprint (for hashing-based algorithm). More details will be presented in Section 4.3.1.

Step 1
Each thread loads 1 ***d***-dimensional reference frame into local memory (***D***-dimensional vector is used to hash the reference frame in case of hashing-based algorithm).

Step 2
Load query frames to shared memory: **each thread** in the block loads **one** d-dimensional query frame to shared memory.

Step 3
Compute the similarity between the reference and all query frames in shared memory using hashing-based or sorting- based algorithms.

Step 4
Save the number of the closest query frame.

Step 5
Repeat 2-5 until no more query frames.

Figure 4.3 Processing steps on GPU

4.2 Clustering-Based Technique

In this section, we describe an efficient search scheme that further reduces the run time. Figure 4.4 shows an illustration of the main components of the proposed technique.

First, the training step partitions reference fingerprints into k different clusters and assigns the closest cluster to each reference fingerprint. Second, a lookup table is built based on query fingerprints and the reference clusters. Finally, the number of matching fingerprints between the query and the reference fingerprints are computed. This section presents these three parts in details.

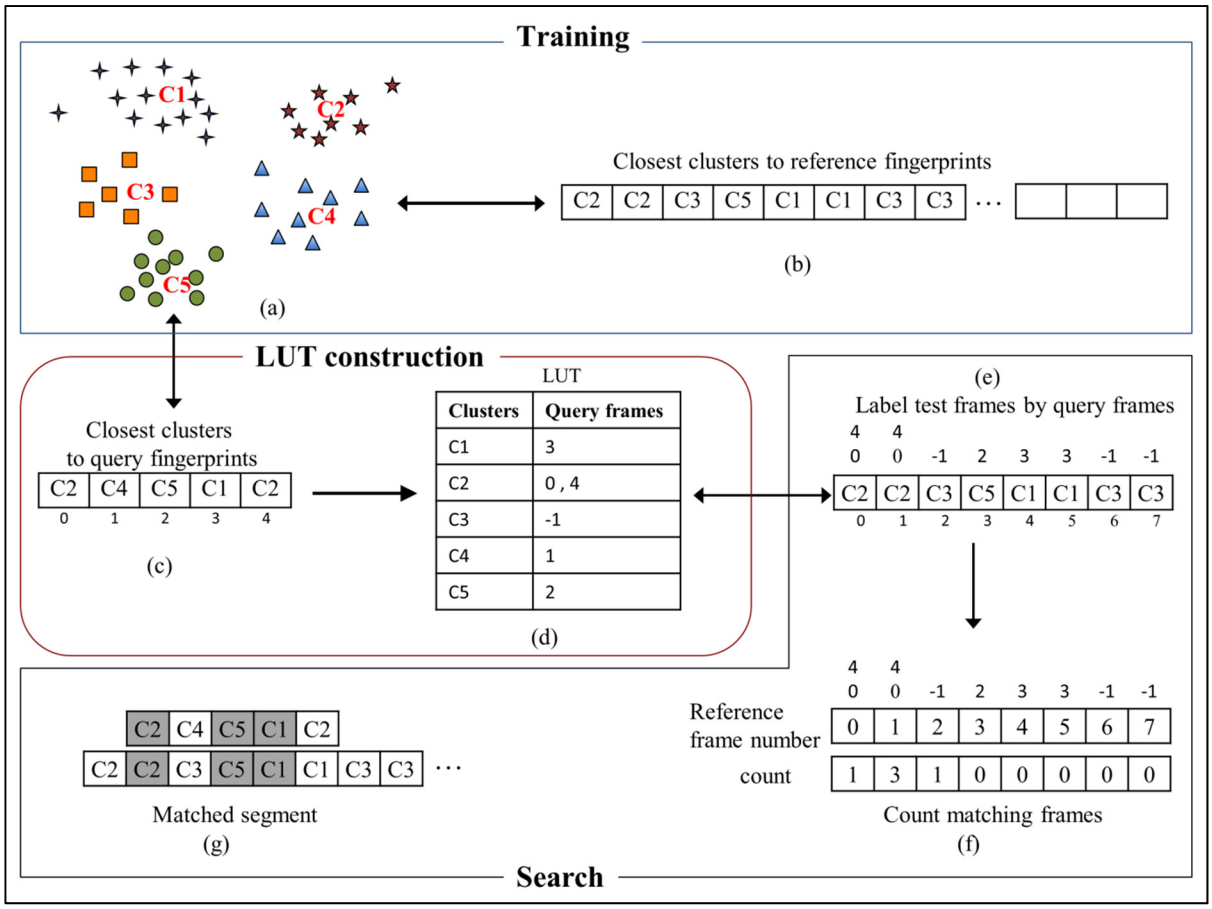

Figure 4.4 Illustration of the main components of the clustering-based technique

4.2.1 Training

The idea behind our approach is to partition the millions of reference fingerprints into smaller groups of similar fingerprints, and represent each group of similar fingerprints by one representative fingerprint. An unsupervised learning algorithm partitions the reference fingerprint space into *k* separate clusters. The centroid of each cluster represents all the fingerprints in that cluster. This clustering reduces the number of comparisons from 61 million to the number of cluster centroids (tens of thousands).

In our clustering experiments, we used k-means like clustering on the reference fingerprints Figure 4.4(a). We use *Forgy* initialization method that randomly chooses *k* fingerprints from

the reference fingerprints as initial centroids. The next step in k-means algorithm assigns each fingerprint to the cluster that has the nearest centroid. Then, it recalculates the new centroids based on the current assignment of the fingerprints to the corresponding clusters. The algorithm continues alternating between reassigning fingerprints to clusters and recalculating the centroids until convergence or reaching a stopping criterion.

To assign a reference fingerprint to a cluster, k-means uses a distance function (e.g. Euclidean distance, Manhattan distance) as a measure of similarity and updates the centroids by computing the cluster means. Our k-means implementation differs from the original algorithm as follows:

- The similarity function is defined as the intersection between two data point elements, since fingerprints encode positions of salient regions.
- The elements of each centroid are defined by looking for the top-d positions shared by all the fingerprints in the cluster, where d is the fingerprint dimension. For example, if there are four fingerprints $P_1 = \{1, 2, 3, 5, 6\}$, $P_2 = \{2, 3, 5, 6, 9\}$, $P_3 = \{2, 5, 6, 11, 15\}$ and $P_4 = \{3, 7, 8, 10, 15\}$ in the cluster C, and $d = 5$, then the new centroid of this cluster C is $= \{2, 3, 5, 6, 15\}$.

Once all the reference fingerprints have been grouped into k separate clusters, we label each reference fingerprint with the cluster number to which it belongs (Figure 4.4(b)).

To sum up, the clustering algorithm takes all reference fingerprints as input, performs a k-means clustering, and returns k centroids and a vector containing cluster indices of each reference fingerprint.

Note that this step is performed offline and only one time for a given reference fingerprints database, and this clustering does not affect the search time for each query.

4.2.2 Lookup Table Construction

For each query, we compute the similarity between each query fingerprint and all the centroids generated in the clustering step. We label each query fingerprint with its closest centroid (Figure 4.4(c)). In this scenario, two similar query and reference fingerprints have a high likelihood of belonging to the same cluster, even if their fingerprints are not identical.

In order to accelerate the search, we construct a lookup table (LUT) where the table index is the cluster number, and the output is the query frame numbers of the query fingerprints that belong to this cluster (Figure 4.4(d)). Note that a −1 value in the LUT means that there is no query fingerprint closest to this cluster (e.g. C3 in Figure 4.4(d)). Each reference frame is then assigned the frame number of the nearest query fingerprint using this lookup table.

4.2.3 Matching Algorithm

The lookup table combined with the search technique illustrated in Figure 2.9 (with SCF-0) accelerates the matching of query fingerprints against the reference fingerprints. For each reference frame, we find the query frame numbers that match the corresponding reference cluster label using the LUT (Figure 4.4(e)).

In other words, we label each reference frame by the query frame numbers that have the same cluster label as the reference cluster label. Then we compute the number of matching frames by updating the count $c(j-i) = c(j-i) + 1$ for each frame j of the reference with label i, where $i > -1$ (i is the query frame number from the LUT corresponding to cluster number of frame j of the reference). An illustration of this search is shown in Figure 4.4(f) where the best count is 3 and is obtained when the first query frame is overlaid on the reference starting with the 2nd frame. Figure 4.4(g) shows this alignment where the matching frames are represented by a grey background.

4.2.4 Two-step Search

The clustering-based technique as described above significantly decreases the computation time by reducing the number of comparison between the query and the reference to the number of clusters instead of the number of all reference fingerprints. However, several transformations (especially those that add irrelevant speech to the query) may make query fingerprints very different from their corresponding reference fingerprints, and therefore belonging to different clusters. This mismatch would decreases the number of matching frames, since a query and a reference fingerprint are considered a match only when they belong to the same cluster.

With the original search (without using the clustering technique), the number of matching frames is computed after labeling each reference frame with the closest query frames (Figure 2.9 with SCF-0). Using the closest frame as a metric between query and reference fingerprints increases the likelihood of matching fingerprints from audio transformed with added speech. Thus, we propose to perform the search in two steps as illustrated in Figure 4.5. In the first step, we use the clustering-based technique to compute and rank the scores of all reference files as described in Figure 4.4. Then, we keep the top-*N* reference files with the highest scores. For each of these top-*N* reference files we extract the best matching segment (i.e. segment that has the highest score). In the second step we rescore these top-*N* reference segments without any clustering as described in Figure 2.9 (with SCF-0), where the nearest neighbor search is performed between query and reference fingerprints. This two-step technique reduces run time significantly while having minimal effect on min NDCR (see Section 4.3.2).

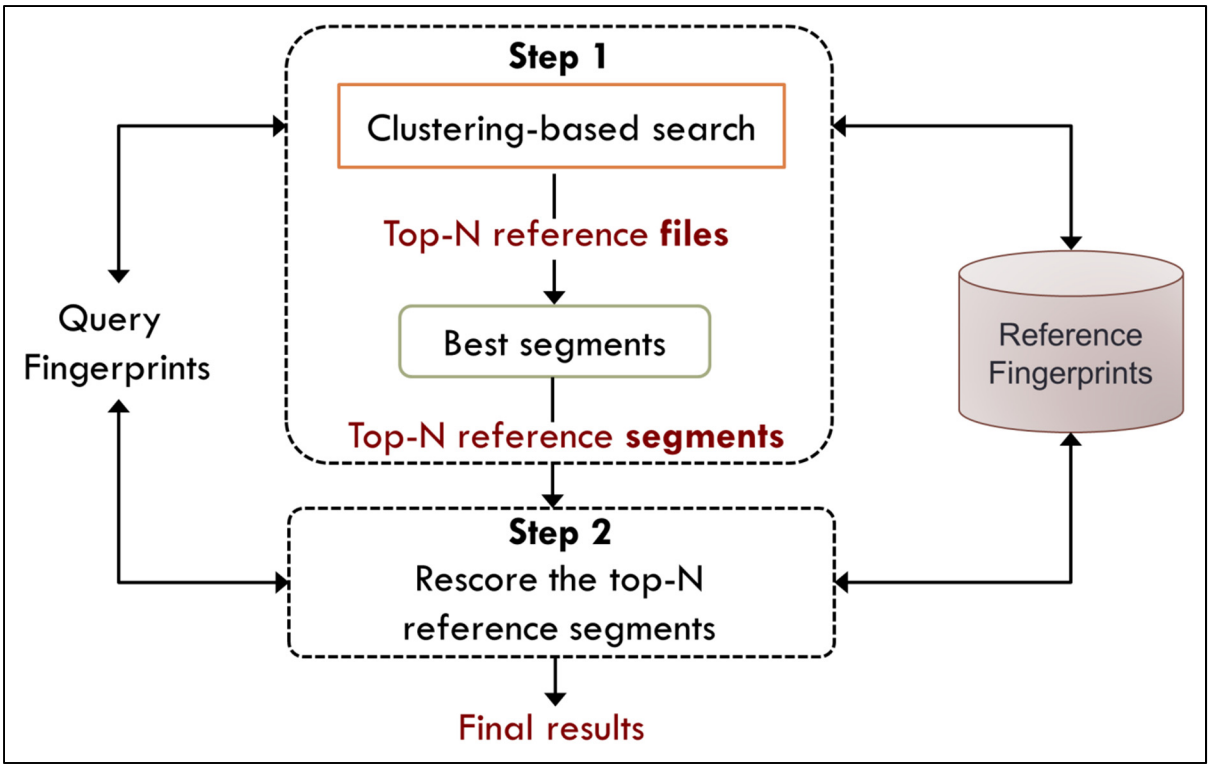

Figure 4.5 Illustration of the two-step search. The Top N results generated from step 1 are rescored in step 2 for accurate results

4.3 Results and Analysis

This section evaluates our system on the well-known TRECVID 2010 CBCD dataset. First, we compare the GPU implementations of the two similarity search algorithms. Then, we use the fastest GPU implementation to optimize the run time versus performance for the clustering-based technique. Next, we compare the proposed system to the *NN-based* (Gupta, Boulianne and Cardinal, 2012), *MASK* (Anguera, Garzon and Adamek, 2012), the implementation of the *energy difference fingerprint* (Haitsma and Kalker, 2002) and the *Shazam* (Wang, 2003) systems (we used the Shazam implementation found in (Ellis, 2009) with the default parameters). We also test the detection performance and the search run time of our system on TRECVID 2009 dataset, and we compare our results to the *WASF* audio features used in (Mou et al., 2013) and the *coherency* vocabulary method proposed in (Douze et al., 2008).

4.3.1 Run Times of GPU Implementations

We run the CPU-based implementation on an Intel(R) Xeon(R) CPU at 2.3 GHz. For the GPU implementation we use the GPU GeForce GTX 580 (based on NVIDIA's Fermi architecture). This GPU has 512 cores and 1.5 GB of global memory.

In the first part of our experiment we evaluate the run time of our copy detection algorithm using all the reference collection and only 10 queries of varying length. Using only 10 queries simplifies testing on different search configurations on the GPU. Finally, we present the best GPU implementation results on all the 1407 queries.

Processing run times (in second) of hashing-based (Algorithm 1) and sorting-based (Algorithm 2) on CPU and GPU using different dimensions of d are presented in Table 4.1. The GPU results are obtained by applying processing steps presented in Figure 4.3 with D = 744, d = {12, 24, and 44}, *nThreads* (number of threads per block) = 256, k (number of query frames to be loaded to shared memory) = *nThreads* and *nBlocks* (number of thread blocks) = number of reference frames/*nThreads*.

Table 4.1 CPU and GPU processing run time in seconds for hashing-based and sorting-based algorithms using 10 queries

	CPU		**GPU**	
d	**Hashing-based**	**Sorting-based**	**Hashing-based**	**Sorting-based**
12	66870	411453	1037	4208
24	132564	895244	2321	11818
44	253148	1690470	15405	54681

From Table 4.1 we can see that hashing-based algorithm is six times faster than sorting-based algorithm on the CPU and more than four times faster on the GPU. This is normal since sorting-based algorithm requires more computing than hashing-based algorithm. The odd thing is that sorting-based algorithm appears to take advantage of GPU parallelism more than

hashing-based algorithm. In fact, hashing-based GPU implementation speeded up the search by 64, 57 and 16 times (for d = 12, 24 and 44, respectively), while the GPU implementation of sorting-based algorithm speeded up the search by 97, 75 and 30 times (for d = 12, 24 and 44, respectively). The reduced speed up with increasing d is due to GPU's shared memory space limitation together with the large D-dimensional vector memory required by the hashing-based algorithm. The following paragraphs confirm this assumption.

Table 4.2 compares run times of hashing-based and sorting-based algorithms on GPU with two configurations: (1) using Shared Memory (SM) for queries as described in Figure 4.3, and (2) without using shared memory for queries. Compared to the first configuration, the threads in the second configuration read query frames directly from the global memory.

Table 4.2 Processing run time using 10 queries of hashing-based and sorting-based algorithms with and without using shared memory

Configuration	**12**	**24**	**44**
Sorting-based without SM	4069	9258	28495
Sorting-based with SM	4208	11818	54681
Hashing-based without SM	1247	2539	4978
Hashing-based with SM	1037	2321	15405

Table 4.2 shows minor performance improvement when using the shared memory with d = 12 and d = 24. However, the use of shared memory with d = 44 increases the run time significantly for both algorithms. Shared memory is very fast compared to the global memory and is supposed to speed up the search by accelerating repetitive access to the query frames. In our case, each block loads k d-dimensional query frames to the shared memory, and if k and/or d are large, the number of concurrent threads decrease (SM may not contain data for the required number of threads). This explains the dramatic drop in performance when using d = 44, especially for hashing-based algorithm. However, the run time for the hashing-based algorithm increases only linearly with d when we do not use the shared memory for queries. This is similar to the linear increase in computing for the CPU implementation of the

hashing-based algorithm. When the threads load the query into the shared memory, the search algorithm becomes almost 7 times slower with d = 44 compared to the same configuration with d = 24. Loading k query frames with d = 44 to shared memory prevents a number of thread blocks to be executed due to the lack of shared memory.

In order to allow maximum number of threads to be executed in parallel, another configuration is to load only one query frame to shared memory (k = 1 instead of k = 256) and to increase the number of threads per block (*nThreads* = 512). This new configuration is more efficient as it minimizes fetching data from global memory while increasing the number of threads per block from 256 to 512.

Table 4.3 shows the run times for this improved configuration (k = 1) for both the sorting-based and hashing-based algorithms. The last row of Table 4.3 shows significant reduction in run time for the hashing-based algorithm when we hash the query frame into a D-dimensional vector instead of the reference frame. The difference between hashing a query frame or a reference frame is the place where the D-dimensional vector is stored (local memory for reference frame and shared memory for query frame). Because the improved configuration loads only one query frame to the shared memory, we have enough shared memory space to run many more threads concurrently. In this case, using shared memory for hashing is better than allocating D-dimensional local memory for this vector.

Table 4.3 Hashing-based and sorting-based run times in seconds with different configurations using 10 queries

		Feature Dimension		
Algorithm	**Configuration**	**12**	**24**	**44**
Sorting-based	k=256	4208	11818	54681
	k=1	1085	8428	20615
Hashing-based	k=256	1037	2321	15405
	k=1 (reference)	1143	2477	5030
	k=1 (query)	444	915	4449

From Table 4.3 we notice a significant reduction in run time when we load one query frame instead of 256 query frames to the shared memory, especially for sorting-based algorithm (improvement of almost 4 times for $d = 12$). Improvement for hashing-based algorithm is significant for $d = 44$ (3 times faster) when we hash the reference into a D-dimensional vector and use $k = 1$ instead of $k = 256$. However, this algorithm with $d = 12$ and $d = 24$ is slightly slower. This is corrected if we hash the query frame into the D-dimensional vector instead of the reference frame (last row of Table 4.3), leading to a two-fold reduction in computing time. This shows that hashing the D-dimensional query vector into shared memory instead of local memory works well. In this situation, only one D-dimensional vector is allocated in the shared memory for each block, otherwise each thread allocates one D-dimensional vector if we hash the query frame into the local memory.

From Table 4.3 we can see that compared to the CPU implementation, the best GPU implementation speeded up the hashing-based algorithm by 150, 144 and 56 times for d equal to 12, 24 and 44, respectively. Similarly, the best GPU implementation speeded up the sorting-based algorithm by 379, 106 and 82 times for d equal to 12, 24 and 44, respectively.

We notice that sorting-based algorithm is more suitable for GPU architecture than the hashing-based algorithm (GPU accelerates sorting-based algorithm by 379 times compared to 150 times for hashing-based algorithm when $d = 12$). This is primarily due to the large D-dimensional vector required for the hashing-based algorithm that makes a number of thread blocks idle.

Although the new configuration (using $k = 1$) reduces significantly the run time for all dimensions (compared to $k = 256$), the run time does not increase linearly, and the hashing-based algorithm is 10 times slower with $d = 44$ compared to $d = 12$. Sorting-based algorithm shows comparable behavior (the run time is increased by a factor of 19 with $d = 44$ compared to $d = 12$) even after organizing efficiently the shared memory.

The reason for this increase in GPU time for $d = 44$ is as follows. In our GPU implementation, each thread loads 1 reference frame into registers (or local memory depending on d). Registers are the fastest memory in the GPU and should be used when possible. However, the GPU has a limited number of registers, and each thread can use only a few of them. The GTX 580 (compute capability 2.0) allows the use of a maximum of 63 32-bit registers per thread, and each multiprocessor has 32K of registers that are partitioned among all resident threads. When the kernel requires a significant number of registers, the compiler uses a mechanism called *register spilling* to minimize the number of used registers. This mechanism implies moving the data out from registers to local memory (after attempting to store the data into L1 and L2 cache memories). Local memory is an abstraction to the local scope of a thread, and the data are in reality stored into global memory; which is very slow compared to registers.

With $d = 44$, there are not enough registers for the launched threads, forcing them to fetch data from global memory. This is not the case for $d = 12$ and $d = 24$, where the data can be stored in registers. The results obtained without using shared memory (see Table 4.2) confirm this. The run time for hashing-based algorithm for $d = 44$ using the optimized configuration (Table 4.3) is only slightly decreased compared to the configuration without using shared memory (Table 4.2). In contrast, the algorithm becomes almost 3 times faster for $d = 12$.

In order to prove that the problem is due to the limited number of registers, we tested the hashing-based algorithm for $d = 44$ when a vector of 12 dimensions (instead of 44) is used to store the reference frames. The detection result is obviously wrong with this modification, however, this experiment gives an estimate of the run time when the similarity is performed with $d = 44$ and only 12 dimensions are loaded into registers. With this configuration, the algorithm takes 1261 seconds to perform the search compared to 4449 seconds. This test confirms that the algorithm would have performed the similarity search in linear time if there were enough registers.

In order to determine the optimal number of dimensions d that ensure a linear execution, we should estimate the number of registers used for each value of d. However, it is not possible to do that without compiling the kernel code. In fact, the compiler together with the assembler performs complex operations to optimize the register usage. To profile register usage, we used the CUDA profiling tool that provides the exact number of registers when compiling the kernel for each value of d.

Figure 4.6 shows the evolution of run times, number of used registers and the amount of used local memory when using varying value of d. It can be seen from Figure 4.6(a) that the run time increases linearly from $d = 12$ to $d = 35$. The algorithm became almost 3 times slower with $d = 36$ compared to $d = 35$. This increase of run time is explained by the fact that the kernel exceeds the allowed number of registers as shown in Figure 4.6(b). The number of registers increase linearly with d and reach its maximum for $d = 35$. The compiler reduces the number of used registers from 60 registers for $d = 35$ to only 39 registers for $d = 36$, and moves the data to local memory. Figure 4.6(c) shows that the kernel does not use any local memory when $d \leq 35$, and the amount of used local memory increases according to d ($d \times 4$ bytes) when $d \geq 36$. From this experiment we can conclude that the hashing-based algorithm can run at linear time when $d \leq 35$.

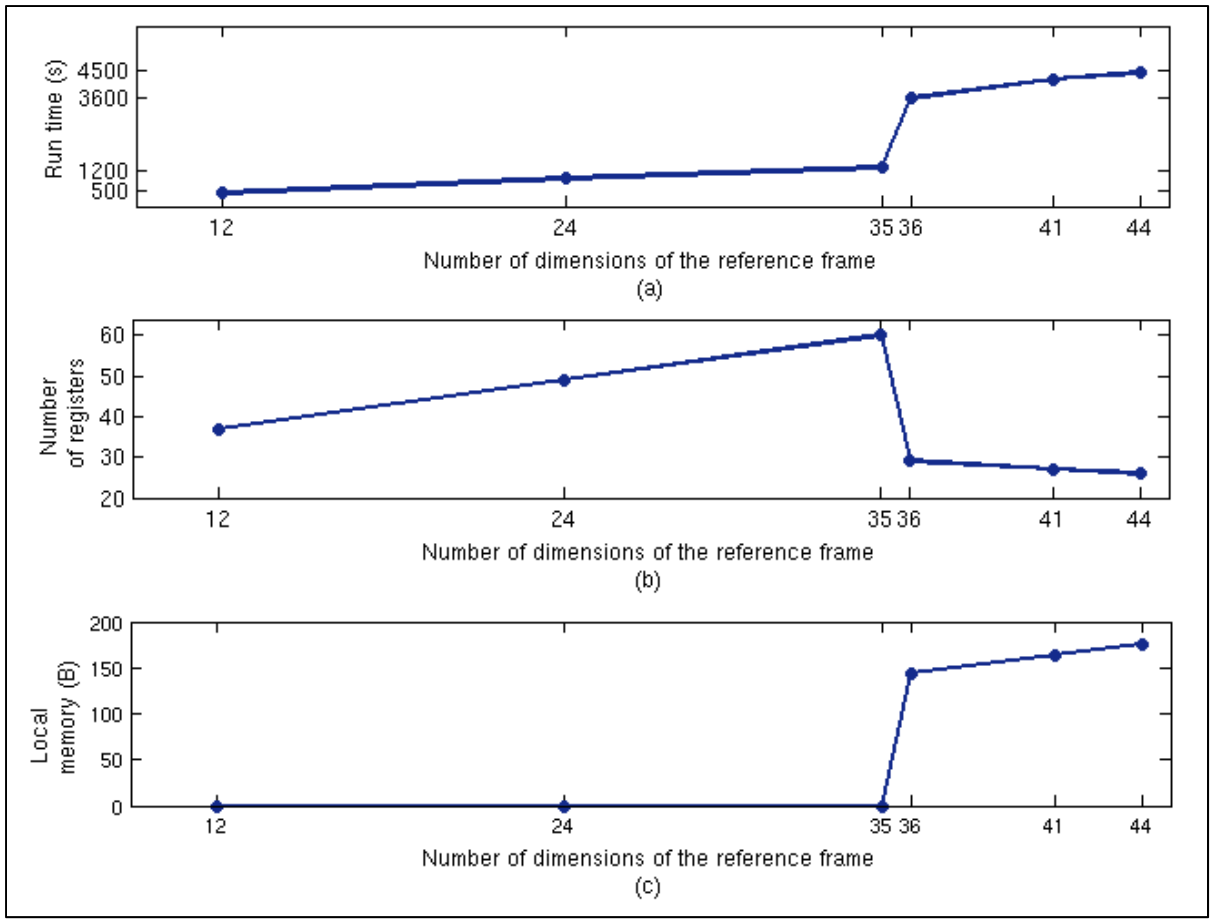

Figure 4.6 Impact of the number of dimensions of the reference frame on (a) run time, (b) number of used registers and (c) amount of used local memory

Besides the problem caused by the limited number of registers, the sorting-based algorithm suffers from *thread divergence serialization*. This problem occurs when threads within a single warp execute different code. The sorting-based algorithm uses an *if-then-else* statement (see Algorithm 2), a common code construct that leads to the divergence of threads. In Algorithm 2, if some threads within a warp evaluate to *true* in the *if* clause, while other threads evaluate to *true* in the *else if* or the *else* clauses, then different code paths will be serialized leading to a significant loss of performance. When d increases, the likelihood that different threads within a warp take different execution paths will increase. This explains the large increase in run time with increasing d for the sorting-based algorithm, and also the

large difference in run time compared to the hashing-based algorithm. It is a good practice to avoid *if clauses* inside thread blocks.

Table 4.4 shows min NDCR, Mean F1 and run time averaged over all transformations (the run time is averaged over all the queries) of our system when processing all the query files (1407 files) using hashing-based algorithm. The rest of the experiments presented in this chapter use hashing-based algorithm since it runs faster than the sorting-based algorithm.

Table 4.4 Proposed system performance using varying dimensions when tested with all the queries on TRECVID 2010

	12	**24**	**44**
Min NDCR	0.149	0.135	0.129
Mean F1	0.903	0.9	0.885
GPU Times (s)	46	95	438

4.3.2 Clustering-based Technique Performance

In this section, we compare the clustering-based technique (CSR) with Salient-Regions (SR, no clustering) for computing and performance tradeoffs.

The number of missed queries (Table 4.5) and the min NDCR (Table 4.6) for each transformation for SR-44 and CSR-44 (44 is the number of dimensions) are shown with grey background. In this experiment, the search with CSR-44 is performed without rescoring the top *N* results (i.e. without the two-step search).

Table 4.5 Number of missed queries by transformation for SR-44 and CSR-44

Method	T1	T2	T3	T4	T5	T6	T7	Total
SR-44	10	10	11	12	21	20	20	104
CSR-44	12	14	28	27	27	44	42	194
CSR-44, *N*=2	12	13	22	20	25	39	36	167

Table 4.6 Min NDCR by transformation for SR-44 and CSR-44

Method	T1	T2	T3	T4	T5	T6	T7	Average
SR-44	0.09	0.09	0.112	0.097	0.172	0.157	0.187	0.129
CSR-44	1.000	1.000	1.000	1.000	0.448	0.679	0.709	0.833
CSR-44, *N*=2	0.09	0.097	0.164	0.149	0.194	0.291	0.269	0.179

The run time per query averaged over all transformations is 3 seconds for CSR-44 compared to 438 seconds for SR-44. Although CSR-44 reduces the run time significantly, it misses more queries than SR-44. Furthermore, scores of many imposter queries are very high in CSR-44, resulting in high decision thresholds (threshold that separates true positives from false alarms) leading to a high min NDCR (Table 4.6). For example, the min NDCR for transform T1 increases from 0.09 for SR-44 to 1 for CSR-44, although CSR-44 missed only two more queries than SR-44 (Table 4.5).

In order to improve the detection performance, we process the query in two steps as described in Section 4.2.4. The number of missed queries and the min NDCR for CSR-44 with the top-2 choices rescored ($N = 2$) is shown in the last rows of Table 4.5 and Table 4.6, respectively.

From Table 4.5, we can see that the two-step search reduces the number of missed queries for almost all the transformations and results in overall 27 less missed queries. Besides, the min NDCR for all the transformations are significantly reduced as can be seen from Table 4.6, and min NDCR is correlated with the number of missed queries, which indicates a good

decision threshold. In addition, using $N = 2$ does not increase the run time, and CSR-44 is still 146 times faster than SR-44.

Figure 4.7 shows the CSR performance tradeoff with increasing N, when using 12 and 44 dimensions. This figure also compares the CSR performance when we rescore the top *N files* and when we rescore the top *N segments.*

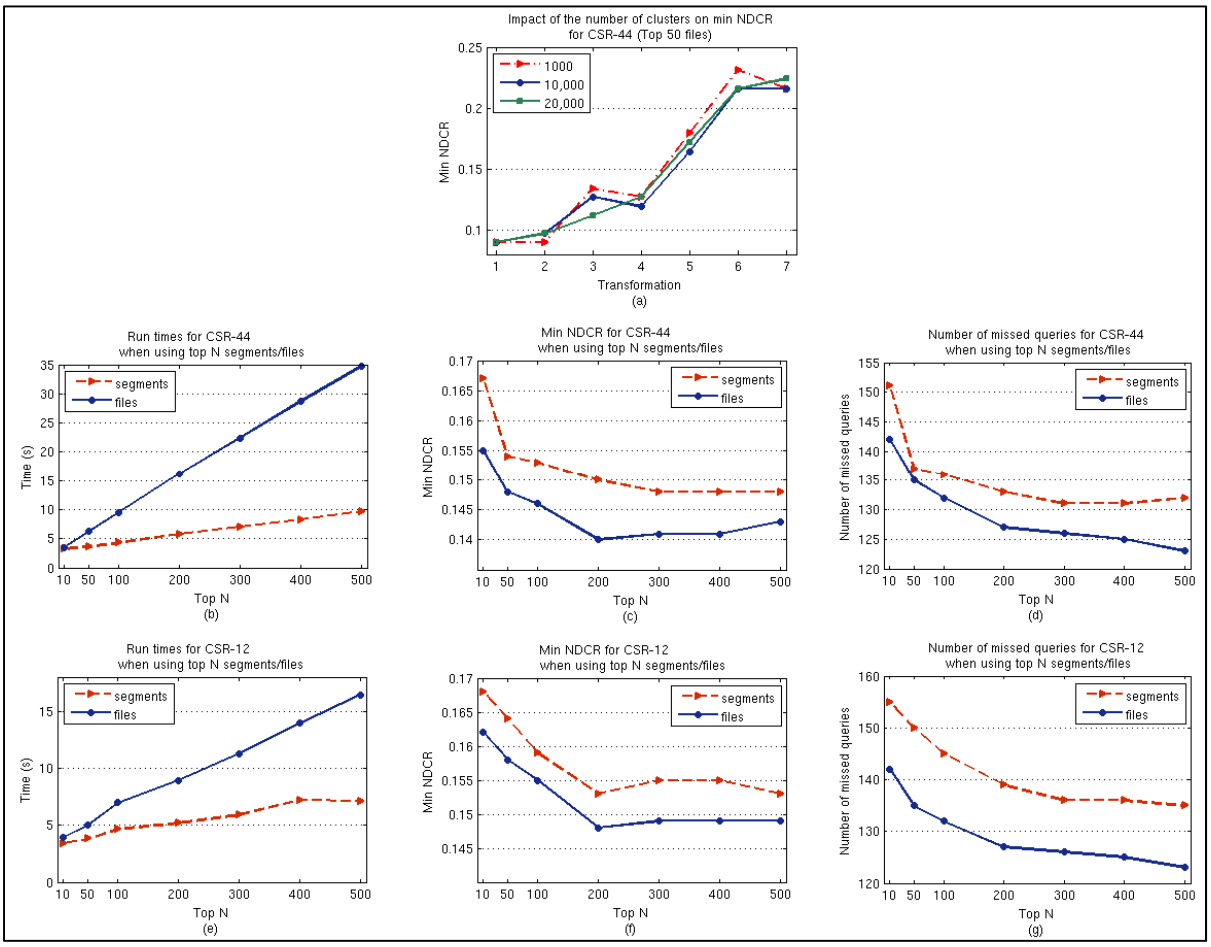

Figure 4.7 Performance of the clustering-based technique: (a) impact of the number of clusters on min NDCR; (b), (c) and (d) evolution of run time, min NDCR and missed queries, respectively, for CSR-44 with Top N files and segments ((e), (f) and (g) for CSR-12)

Figure 4.7(a) shows the min NDCR for each transformation with $d = 44$ and $N = 50$ files for varying number of clusters. From this figure, it can be seen that using 1,000 clusters gives the highest min NDCR for almost all the transformations (i.e. the worst result). Results of min NDCR obtained with 10,000 and 20,000 clusters are very close. The average min NDCR over all transformations is equal to 0.147 when using 10,000 clusters compared to 0.148 when using 20,000 clusters. Since 10,000 clusters gave the best results, the rest of the experiments are performed using this number of clusters.

We see a significant run time reduction when the top N *segments* are rescored instead of the top N *files* for $d = 12$ (see Figure 4.7(e))) and $d = 44$ (see Figure 4.7(b))). The run time for d equals 44 increases by only 6 seconds from $N = 10$ segments to $N = 500$ segments compared to 31 seconds when the number of files increase from 10 to 500. Also, the run time difference between $d = 12$ and $d = 44$ becomes less important when rescoring segments instead of files. In fact, the run time is almost identical for both the dimensions when $N < 300$ segments. The clustering-based technique reduces the run time difference between $d = 12$ and $d = 44$. Without clustering, the search algorithm becomes 10 times slower when d increases from 12 to 44 (Table 4.4).

From Figure 4.7(d) we can see that the number of missed queries decrease as N increases for $d = 44$. However, the top N files result in fewer missed queries than the top N segments (similar result is obtained with $d = 12$ as shown in Figure 4.7(g)). The same can also be said for min NDCR (averaged over all transformations) for $d = 12$ (see Figure 4.7(f)) and $d = 44$ (see Figure 4.7(c))). However for some cases, the min NDCR becomes higher with increasing N, even though more queries are correctly detected. For example, with $N = 200$ files, the system missed 127 queries and gave a min NDCR of 0.140 (see Figure 4.7(c) and Figure 4.7(d), respectively). When N is increased to 500 files the number of missed queries is reduced to 123, yet the min NDCR increases to 0.143. This can be explained by the fact that with increasing N, the system not only detects correctly more queries, but may also give high scores for some imposters, resulting in a high decision threshold. Thus, the optimal number N corresponds to the result with the lowest min NDCR. For $d = 44$ the best results are achieved

with $N = 200$ files or $N = 300$ segments (see Figure 4.7(c)). The lowest min NDCR for $d = 12$ is obtained with $N = 200$ files or segments (see Figure 4.7(f)).

Table 4.7 compares the min NDCR for each audio transformation obtained with SR (no clustering) and with CSR (rows with grey background) using both 12 dimensions ($N =$ 200 files or segments) and 44 dimensions ($N = 200$ files or 300 segments). The last two rows of Table 4.7 show the min NDCR of NN-based (Gupta, Boulianne and Cardinal, 2012) and Shazam (Wang, 2003) audio fingerprinting systems.

Table 4.7 Min NDCR of NN-based, Shazam, Salient Regions (SR) and the best results achieved using the clustering technique (CSR) with 12 and 44 dimensions

Method	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**Avg. min NDCR**	**Avg. time (s)**
SR-12	0.104	0.112	0.149	0.112	0.179	0.187	0.201	0.149	46
CSR-12 Top 200 files	0.097	0.097	0.134	0.112	0.179	0.209	0.209	0.148	9
CSR-12 Top 200 segments	0.097	0.104	0.149	0.119	0.187	0.209	0.209	0.153	5
SR-44	0.09	0.09	0.112	0.097	0.172	0.157	0.187	0.129	438
CSR-44 Top 200 files	0.09	0.097	0.097	0.119	0.164	0.209	0.201	0.140	16
CSR-44 Top 300 segments	0.09	0.097	0.127	0.127	0.172	0.209	0.216	0.148	7
NN-based	0.179	0.187	0.194	0.187	0.201	0.194	0.209	0.193	178
Shazam	0.373	0.366	0.507	0.47	0.515	0.597	0.59	0.488	1.5

From Table 4.7 we can see that SR-44 gives the lowest min NDCR averaged over all transformations. Using clustering increases the min NDCR from 0.129 (for SR-44) to 0.140 (for CSR-44 with top 200 files); however, CSR-44 speeds up the search by 27 times. The clustering-based technique seems to work better for $d = 12$ and $N = 200$ files, reducing the min NDCR from 0.149 to 0.148 (compared to SR-12) while being 5 times faster.

On the other hand, using the top N segments instead of files increases the min NDCR for most of the transformations. However, it speeds up the search by 9 and 62 times (d = 12 and d = 44, respectively) compared to SR-12 and SR-44, respectively.

On the other hand, the min NDCR averaged over all transformations of 0.129 achieved with our system SR-44 is the best result compared to the other two systems. In fact, SR-44 gave the lowest min NDCR for all transformations compared to NN-based which gave a min NDCR averaged over all transformations of 0.193, and to Shazam system that gave the highest min NDCR averaged over all transformations of 0.488. However, Shazam is the fastest system and requires less than 2 seconds, on average, to process a query. NN-based system takes roughly 178 seconds which makes it 4 times slower than SR-12. SR-44 is the slowest system and takes more than 430 seconds on average to process a query. However, the clustering based technique reduces the run time with a slight loss of performance.

Although CSR reduces the detection performance compared to SR, it still gave lower min NDCR compared to NN-Based and Shazam systems. Furthermore, our system achieved better detection performance (min NDCR of 0.179) compared to these two systems even with CSR-44 with N = 2 (see Table 4.6), while being 60 times faster than the NN-based system.

Besides the NN-based and Shazam systems, our system outperforms other systems such as MASK (Anguera, Garzon and Adamek, 2012) and the implementation of the energy difference fingerprint (Haitsma and Kalker, 2002). On TRECVID 2010 dataset, MASK achieved a min NDCR averaged over all transformations of 0.43 compared to 0.55 achieved by the energy difference fingerprint (these results are published in (Anguera, Garzon and Adamek, 2012)). Even CSR-12 with Top 200 segments (see Table VI) gave a significantly lower min NDCR of 0.153.

4.3.3 Validation Results on TRECVID 2009

In order to validate the copy detection results obtained on TRECVID 2010 dataset, we tested our system on TRECVID 2009 dataset with the same parameters. For the TRECVID 2009, we do not generate additional fingerprints with varying speeds. Table VII shows min NDCR for SR-44, CSR-44 (with Top 300 segments), WASF (Mou et al., 2013) and for the coherency vocabulary method (Douze et al., 2008).

Table 4.8 Min NDCR for SR-44 and CSR-44 compared to the WASF and Coherency method on TRECVID 2009 dataset

Method	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**Average min NDCR**	**Average times (s)**
SR-44	0.06	0.067	0.082	0.075	0.09	0.157	0.127	0.094	214
CSR-44 Top 300 segments	0.082	0.09	0.119	0.097	0.119	0.194	0.224	0.132	3
WASF	0.090	0.09	0.09	0.09	0.194	0.172	0.187	0.130	54
Coherency	0.142	0.306	0.629	0.485	0.530	0.697	0.918	0.529	8

From Table 4.8, we can see that SR-44 shows good detection results and achieves the lowest min NDCR for all the transformations. Although the averaged min NDCR of CSR-44 is 0.132 compared to 0.094 for SR-44, CSR-44 speeded up the search by 71 times compared to SR-44. In addition, CSR-44 achieves comparable performance to WASF while being 18 times faster. The coherency vocabulary method is 7 times faster than WASF, but results in relatively poor detection performance (averaged min NDCR of 0.529).

4.3.4 Scalability

To test the detection performance of our system on a larger reference dataset, we doubled the reference dataset by combining TRECVID 2009 and 2010 reference datasets. The evaluation

of our system using TRECVID 2010 queries shows that doubling the size of the reference dataset slightly impacts the detection performance of our system (average min NDCR of 0.130 with the new dataset compared to 0.129 achieved on TRECVID 2010 references only for SR-44). Similarly, the average min NDCR of CSR-44 with Top 300 segments goes from 0.148 using TRECVID 2010 only, to 0.150 with the doubled dataset.

In addition, Table 4.9 compares the run times of our system on the doubled reference dataset when using two different GPUs: GTX 580 and GTX Titan X.

Table 4.9 Average run time (secs) on GTX 580 and GTX TiTan X for SR-44 and CSR-44 top 300 segments on the doubled reference dataset

Device	**SR-44**	**CSR-44 Top 300 segments**
GTX 580	872	9
GTX Titan X	153	6

Table 4.9 shows that doubling the size of the reference dataset leads to a linear increase of the run time for SR-44 (from 438 secs to 872 secs) and sublinear increase for CSR-44 Top 300 segments (from 7 secs to 9 secs). GTX Titan X speeded up the search for SR-44 by almost 6 times compared to GTX 580. We also notice a small speed improvement for CSR-44 Top 300 segments when using *GTX Titan X*.

4.3.5 Shazam versus CSR-44

The question now is how we can reduce the run time of our system to match the run time of the Shazam system. The average run time of CSR-44 with top $N = 2$ files is 3 seconds per query. In fact, with this configuration the similarity search on the GPU is performed with almost no cost, and all the processing time is spent in rescoring on the CPU. With the two-step clustering-based search, the matching algorithm is executed twice: the first time to count the number of matching *clusters* (first step), and the second time to count the number of

matching *frames* between the query and the reference (second step). Counting the number of matching clusters is slower than counting the number of matching frames for two reasons. First, in the second step of the retrieval algorithm we process only *N* reference files/segments compared to all reference files in the first step. Second, a reference frame can be labeled with multiple query frames (when several query fingerprints belong to the same cluster, see Figure 4.4(e)) compared to only one query frame (i.e. the closest query frame to the reference) for the second matching step (see Figure 2.9).

In order to reduce the run time, we propose to keep only one query frame when we perform the first matching algorithm instead of multiple frames. In the example of Figure 4.4(d), this implies keeping only the query frame number 0 instead of 0 and 4 in the LUT where the input is C2.

When we tested the modified matching algorithm using CSR-44 with top 10 segments, our system became slightly faster than Shazam with an average processing time of 1.3 seconds, which is also 137 times faster than the NN-based system. Understandably, the detection performance decreased, and the system gave an average min NDCR of 0.193 (same as NN-based system), which is far better than the detection performance achieved by Shazam (0.488).

Note that the 1.3 second runtime includes the similarity search on GPU and two matching algorithms (first and second steps in Section 4.2.4). The matching algorithms are fast due to 2 reasons: The first matching algorithm in CSR-44 does not process many reference frames (i.e. there is -1 in the LUT). In addition, the second matching algorithm processes fewer reference frames: frames corresponding to only 10 reference segments.

On the other hand, we get better localization accuracy than the NN-based and the Shazam systems as shown in Table 4.10 (higher Mean F1 means better localization). Our method gave Mean F1 averaged over all transformations of 0.911, compared to 0.807 and 0.693 for

Shazam and NN-based systems, respectively. Note that our system gave similar localization accuracy regardless of the configuration (SR-*x*, CSR-*x* using top *N* files or segments).

Table 4.10 Mean F1 for the proposed system (CSR-12 with Top 200 segments), NN-based and Shazam systems

System	T1	T2	T3	T4	T5	T6	T7	Average
CSR-12	0.923	0.935	0.925	0.923	0.892	0.898	0.883	0.911
NN-Based	0.685	0.695	0.701	0.691	0.685	0.691	0.703	0.693
Shazam	0.765	0.789	0.825	0.795	0.803	0.841	0.834	0.807

Note that the average run time of the CPU implementation of SR-44 is estimated to be more than 7 hours. The average run time of 1.3 seconds achieved by the proposed two-step search makes our algorithm 19472 times faster than the CPU implementation without clustering.

4.4 Summary

The Salient-Regions fingerprint introduced in Chapter 2 (for audio) and Chapter 3 (for video) have shown their robustness against complicated audio and video transformations. Each fingerprint generated either from the video or the audio signal encodes the positions of salient regions selected from either binary image (for audio) or greyscale image (for video). The similarity between two fingerprints is defined as the intersection between their elements. Because of the high dimensionality of the fingerprints and the large volume of data, searching over millions of fingerprints is computationally challenging.

We have investigated in this chapter the use of the GPU to reduce the computing. More precisely, we have described an efficient way of utilizing the GPU memories to perform a similarity search in parallel on a large database of fingerprints. Experimental results demonstrate that the GPU implementation can accelerate the search by over hundred times compared to the CPU implementation. We have also shown how to optimize this GPU implementation for fingerprints with large dimensions ($d = 44$).

To reduce the run time even further, we have proposed a two-step clustering based search. In the first step, we cluster the reference fingerprints into several thousand clusters, reducing the nearest-neighbor search time significantly. In the second step, we rescore the top N results obtained in the first step to produce more accurate copy detection. This two-step strategy works very well and achieved a speed up of 146 times while increasing the min NDCR from 0.129 to 0.179. Basically, clustering creates a tradeoff between the speed and the detection accuracy. The experimental results have shown that using $d = 44$ and top 200 files accelerates the search 27 times and achieves a min NDCR of 0.140. For $d = 12$, the clustering-based technique not only speeds up the search, but also results in better detection performance (compared to SR-12).

In addition, clustering allowed us to match the speed of the Shazam system, and at the matched speed, our system outperformed Shazam by a significant margin (min NDCR of 0.193 versus 0.488). Our system also outperforms NN-based system in terms of min NDCR, Mean F1 and run time.

CONCLUSION

In this thesis, we have addressed the problem of multimedia content-based copy detection, which is a very critical challenge for many tasks such as the protection of copyrighted content. Our contributions in this area are mainly related to two important requirements of any multimedia copy detection system: robustness and efficiency. In particular, we have introduced a novel approach to extract audio features from the spectrogram that allows the generation of three different fingerprints: Global Mean, Local Mean and Salient-Regions fingerprints. We have also described two new visual feature extraction approaches that generate V-intensity and V-motion fingerprints. In addition, we have proposed two solutions to accelerate the search of fingerprints in context of large-scale audio/video fingerprint database.

The idea behind the design of our Global Mean and Local Mean fingerprints is that the spectrograms of an original audio and its copy look very similar. However, distortion may change visual information in the spectrogram. To reduce audio mismatch due to these distortions, we have first converted the spectrogram into 2-D binary images using the global or local mean of the spectral values in the spectrogram. Then, we have generated different versions of fingerprints by keeping an incremental amount of signal information based on a spectral energy threshold for each version. With this strategy, one of the spectrogram versions of the copy is more likely to have spectrogram similar to the original.

Each Global Mean and Local Mean fingerprint version are represented by a d-dimensional vector obtained by dividing the binary image into d slices and each element of this vector represents the sum of each slice. During retrieval, each reference fingerprint is labeled with the closest query fingerprint. Then, we count the number of matching frames for each alignment between query and reference fingerprints. We have improved this search by associating the nearest query frame and the N nearest successive neighboring frames of the query when counting the number of matching frames.

We have shown that Global Mean and Local Mean fingerprints are robust against different audio transformations when evaluated on TRECVID 2009 and 2010 datasets. In addition, we have demonstrated that combining results from Global Mean and Local Mean improves the average min NDCR by 18% compared to the best results achieved using these features separately.

Salient-Regions fingerprints are also extracted from binary images by using the global mean as a threshold. However unlike Global Mean and Local Mean fingerprints, we have generated only one fingerprint version with Salient-Regions to reduce the search algorithm run time. In addition, Salient-Regions fingerprint representation is performed in a different way. First, we divide the binary image into D tiles and we compute the sum in each tile. Then, we number each tile of the image from 1 to D, and then we look for the d tiles that have the highest values. The Salient-Regions fingerprint encodes the positions of these d tiles.

We have evaluated the Salient-Regions on TRECVID 2009 and 2010 datasets. We have shown that Salient-Regions fingerprint outperforms both fingerprints. Indeed, it decreases min NDCR averaged over all transformations by 29% and 42% compared to Global Mean and Local Mean respectively. In addition, we have compared our system to two state-of-the-art audio copy detection systems. Results of this comparison have shown that our system outperformed both systems for all transformations on TRECVID 2010. In fact, the min NDCR averaged over all transforms obtained by our system is equal to 0.129 compared to 0.193 and 0.488 achieved by NN-based and Shazam respectively. Besides, the best localization accuracy is achieved with the proposed Salient-Regions fingerprint with Mean F1 averaged over all transformations of 0.885 compared to 0.807 and 0.693 for Shazam and NN-based systems respectively. In TRECVID 2009 dataset, our system outperformed by far Shazam system that missed 845 queries compared to only 76 queries missed by our system. The min NDCR averaged over all transformation achieved by the proposed system is equal to 0.09 which is comparable to the 0.06 achieved by NN-based system.

For the video part, we have applied the same strategy used to generate audio Salient-Regions fingerprints by encoding the positions of salient regions. However, instead of selecting the salient regions from binary images, we have selected video-based salient regions from grayscale video images. We have proposed two methods based on this approach that generate V-intensity and V-motion fingerprints. For these two fingerprint extraction methods, we apply a tile of fixed size and compute the sum of the pixel values in each small square of the tile. V-intensity selects from each video image the *top-d* regions that have the average intensity values. V-motion selects the regions that have the highest variations compared to the same regions in the previous frame.

We have evaluated the video fingerprints on TRECVID 2009 and TRECVID 2010 datasets. We have shown their robustness against different video transformations. In fact, the proposed fingerprints achieved excellent results for all transformations that do not include geometric transformations. On TRECVID 2009 dataset for example, V-motion correctly detects all queries for three transformations without any false alarm (i.e. min NDCR = 0). We have also shown that these video fingerprints results in better performance for many transformations when compared to NN-based, DCT and DC-SIFT features.

To address the problem of detecting a video copy transformed by audio and video transformations, we have proposed a simple fusion strategy that combines the results achieved by the audio Salient-Regions fingerprint and the video Salient-Regions fingerprints. We have shown that our system achieved excellent results for this task with a min NDCR averaged over all transformation of 0.021 and 0.053 for TRECVID 2009 and 2010 datasets, respectively.

Regarding the efficiency requirement, we have proposed two solutions that considerably improve the speed of the search of audio/video fingerprints. The first solution is based on the use of a GPU to parallelize the similarity search algorithm. We have implemented two similarity search algorithms on GPU. We have described an efficient parallel design optimized for the search of fingerprints in a large reference dataset. We have shown that

appropriate use of the GPU memory space that maximizes the number of concurrent threads has a significant impact on the overall compute time when using fingerprints of varying dimensions. With simple modifications we have obtained up to 4 times better GPU run time when using GPU memory to maximize concurrent threads.

Compared to the CPU only implementations, the proposed GPU implementation reduced the run times by up to 150 times for one intersection algorithm and by up to 379 times for the other intersection algorithm.

On the other hand, we have proposed a two-level search algorithm that further reduces the run time. In the first step, we compute the number of matching frames between the query and the reference fingerprints based on a clustering technique combined with a lookup table. In the second step, we rescore the top N results to produce more accurate copy detection. This technique achieved remarkable speed improvements and allowed our system to match the Shazam speed system while achieving significantly better detection performance. Besides, the proposed system achieved better results than the robust NN-based system while being 137 times faster.

Future work

In this thesis, we have proposed a multimedia content-based copy detection system that is highly robust to a variety of audio and video transformations. However, our audio fingerprints are not invariant to time-frequency scale modifications. In this work, we have proposed to generate multiple query fingerprints by changing the sampling frequency of the audio to cope with the time-frequency scale variation between the query and the reference. This strategy worked well with small time-frequency scale variations, but it does not detect all the time-frequency scale transformed copies especially those that have undergone large time-frequency scale modifications. Thus, future work will be mainly devoted to explore new strategies to make the Salient-Regions audio fingerprint invariant to time-frequency scale modifications.

A possible way to achieve this goal is to encode the positions of the selected salient regions relative to each others, instead of their positions within the window. A time-frequency modification applied to an audio signal leads to a proportional change in the time and frequency axes. Thus, this proposed strategy will ensure that the temporal and the spatial information will not be included in the fingerprint.

On the other hand, the proposed video fingerprints have shown excellent detection performance for queries that do not include geometric transformations. However, the performance of our video fingerprinting system decreases with transformations that change the content of the video image like the crop transformation. In fact, such transformations change the positions of the salient regions of the video image and prevent our system to detect the query accurately. Thus, we need to make the V-intensity and V-motion video fingerprints invariant to content-altering video transformations that change the positions of the selected salient regions. In general, global visual features are usually sensitive to content-changing transformations, in contrast to local visual features that are robust to such types of transformations. In future work, we plan to improve the robustness of our system by generating additional local fingerprints and combining them to V-intensity and V-motion fingerprints.

In this work, we described a PiP detection algorithm that detected up to 79% of the inserted PiPs on TRECVID 2010 dataset. In this algorithm we used *Roberts* method to detect edges, and for many queries this method failed to detect candidates PiPs edges. In future work, we suggest to test this technique using other edge detection methods such as *Canny* edge detector, and to try with different parameters in order to improve the PiP detection performance.

As another part of our future work, we will investigate the use of other kind of clustering techniques. Recall that in this study, we employed k-means like clustering technique to divide the reference fingerprint space into k separate clusters. To determine the ideal number of clusters, we tested our system using different values of k. This operation should be

repeated every time new fingerprints are added to the reference dataset. Thus, we should explore other clustering techniques that do not need the number of clusters to be specified.

Finally, we intend to explore in depth the parameter setting of our system (number of salient regions, size of tiles, window size, frame advance, etc.) in order to identify the combination of parameter settings that ensures the best performances in terms of memory usage, detection performance, precision accuracy and run time.

LIST OF REFERENCES

Anguera, Xavier, Antonio Garzon and Tomasz Adamek. 2012. « MASK: Robust local features for audio fingerprinting ». In *2012 13th IEEE International Conference on Multimedia and Expo, ICME 2012, July 9, 2012 - July 13, 2012*. (Melbourne, VIC, Australia), p. 455-460. IEEE Computer Society. < http://dx.doi.org/10.1109/ICME.2012.137 >.

Audiblemagic. < http://www.audiblemagic.com/ >. Consulté le 1 septembre 2015.

Awad, George, Paul Over and Wessel Kraaij. 2014. « Content-based video copy detection benchmarking at TRECVID ». *ACM Transactions on Information Systems (TOIS),* vol. 32, nº 3, p. 14.

Ayari, Mohamed, Jonathan Delhumeau, Matthijs Douze, Hervé Jégou, Danila Potapov, Jérôme Revaud, Cordelia Schmid and Jiangbo Yuan. 2011. « INRIA@TRECVID'2011: Copy Detection & Multimedia Event Detection ». In *TRECVID workshop*.

Baluja, S., and M. Covell. 2007. « Audio fingerprinting: combining computer vision data stream processing ». In *2007 IEEE International Conference on Acoustics, Speech, and Signal Processing, 15-20 April 2007*. (Piscataway, NJ, USA), p. 213-16. Coll. « 2007 IEEE International Conference on Acoustics, Speech, and Signal Processing (IEEE Cat. No. 07CH37846) »: IEEE.

Baluja, Shumeet, and Michele Covell. 2008. « Waveprint: Efficient wavelet-based audio fingerprinting ». *Pattern recognition,* vol. 41, nº 11, p. 3467-3480.

Bellettini, Carlo, and Gianluca Mazzini. 2010. « A framework for robust audio fingerprinting ». *Journal of Communications,* vol. 5, nº 5, p. 409-424.

Bhat, Dinkar N., and Shree K. Nayar. 1998. « Ordinal measures for image correspondence ». *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 20, nº 4, p. 415-423.

Bilal Orhan, O., Jingen Liu, Jason Hochreiter, Jonathan Poock, Qinfeng Chen, Ajay Chabra and Mubarak Shah. 2008. « University of Central Florida at TRECVID 2008 content based copy detection and surveillance event detection ». In *TREC Video Retrieval Evaluation, TRECVID 2008, November 17, 2008 - November 18, 2008*. (Gaithersburg, MD, United states), p. National Institute of Standards and Technology (NIST). Coll. « 2008 TREC Video Retrieval Evaluation Notebook Papers »: National Institute of Standards and Technology.

Bosch, Anna, Andrew Zisserman and Xavier Munoz. 2008. « Scene classification using a hybrid generative/discriminative approach ». *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 30, nº 4, p. 712-727.

Cano, P., E. Batle, T. Kalker and J. Haitsma. 2002. « A review of algorithms for audio fingerprinting ». In *2002 IEEE 5th Workshop on Multimedia Signal Processing, 9-11 Dec. 2002*. (Piscataway, NJ, USA), p. 169-73. Coll. « Proceedings of 2002 IEEE Workshop on Multimedia Signal Processing (Cat. No.02TH8661) »: IEEE.

Cano, Pedro, Eloi Batlle, Ton Kalker and Jaap Haitsma. 2005. « A review of audio fingerprinting ». *Journal of VLSI signal processing systems for signal, image and video technology,* vol. 41, nº 3, p. 271-284.

Cardinal, Patrick, Vishwa Gupta and Gilles Boulianne. 2010. « Content-based advertisement detection ». In *Eleventh Annual Conference of the International Speech Communication Association*.

Carpentier, Grégoire. 2005. « Information technology—Multimedia content description interface—Part 4: Audio, AMENDMENT 2: High-level descriptors ». In *Motion Picture Expert Group (ISO/IEC JTC 1 SC29*. p. -.

Chandrasekhar, Vijay, Matt Sharifi and David A Ross. 2011. « Survey and Evaluation of Audio Fingerprinting Schemes for Mobile Query-by-Example Applications ». In *ISMIR*. Vol. 20, p. 801-806.

Chen, Jianping, and Tiejun Huang. 2008. « A robust feature extraction algorithm for audio fingerprinting ». In *9th Pacific Rim Conference on Multimedia, PCM 2008, December 9, 2008 - December 13, 2008*. (Tainan, Taiwan). Vol. 5353 LNCS, p. 887-890. Springer Verlag. < http://dx.doi.org/10.1007/978-3-540-89796-5_106 >.

Chen, Li, and F. W. M. Stentiford. 2008. « Video sequence matching based on temporal ordinal measurement ». *Pattern Recognition Letters,* vol. 29, nº 13, p. 1824-1831.

Ching-Yung, Lin, and Chang Shih-Fu. 2001. « A robust image authentication method distinguishing JPEG compression from malicious manipulation ». *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 11, nº 2, p. 153-68.

Chiu, Chih-Yi, Tsung-Han Tsai, Yu-Cyuan Liou, Guei-Wun Han and Hung-Shuo Chang. 2014. « Near-duplicate subsequence matching between the continuous stream and large video dataset ». *Multimedia, IEEE Transactions on,* vol. 16, nº 7, p. 1952-1962.

Chou, Chien-Li, Hua-Tsung Chen and Suh-Yin Lee. 2015. « Pattern-Based Near-Duplicate Video Retrieval and Localization on Web-Scale Videos ». *Multimedia, IEEE Transactions on,* vol. 17, nº 3, p. 382-395.

Dailymotion. < - www.dailymotion.com >. Consulté le 1 septembre 2015.

Disney. < http://disney.com/ >. Consulté le 1 septembre 2015.

Douze, Matthijs, Adrien Gaidon, Herve Jegou, Marcin Marszałek and Cordelia Schmid. 2008. « INRIA-LEARs video copy detection system ». In *TREC Video Retrieval Evaluation (TRECVID Workshop)*.

Douze, Matthijs, Herve Jegou and Cordelia Schmid. 2010. « An image-based approach to video copy detection with spatio-temporal post-filtering ». *IEEE Transactions on Multimedia,* vol. 12, nº 4, p. 257-266.

Duda, R. O., and P. E. Hart. 1972. « Use of the Hough transformation to detect lines and curves in pictures ». *Communications of the ACM,* vol. 15, nº 1, p. 11-15.

Ellis, Dan. 2009. « Robust landmark-based audio fingerprinting ». *Online Serial],(2009 May), Available at HTTP: http://labrosa. ee. columbia. edu/~ dpwe/resources/matlab/fingerprint,* vol. 4.

Esmaeili, Mani Malek, Mehrdad Fatourechi and Rabab Kreidieh Ward. 2011a. « A robust and fast video copy detection system using content-based fingerprinting ». *IEEE Transactions on Information Forensics and Security,* vol. 6, nº 1, p. 213-226.

Esmaeili, Mani Malek, Mehrdad Fatourechi and Rabab Kreidieh Ward. 2011b. « A robust and fast video copy detection system using content-based fingerprinting ». *Information Forensics and Security, IEEE Transactions on,* vol. 6, nº 1, p. 213-226.

Facebook. < www.facebook.com >. Consulté le 1 septembre 2015.

Garcia, Vincent, Eric Debreuve and Michel Barlaud. 2008. « Fast k nearest neighbor search using GPU ». In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*. p. 1-6. IEEE.

Gupta, V. N., G. Boulianne and P. Cardinal. 2012. « CRIM's content-based audio copy detection system for TRECVID 2009 ». *Multimedia Tools and Applications,* vol. 60, nº 2, p. 371-87.

Gupta, Vishwa, Parisa Darvish Zadeh Varcheie, Langis Gagnon and Gilles Boulianne. 2012. « Content-based video copy detection using nearest-neighbor mapping ». In *2012 11th International Conference on Information Science, Signal Processing and their Applications, ISSPA 2012, July 2, 2012 - July 5, 2012*. (Montreal, QC, Canada), p. 918-923. Coll. « 2012 11th International Conference on Information Science, Signal Processing and their Applications, ISSPA 2012 »: IEEE Computer Society. < http://dx.doi.org/10.1109/ISSPA.2012.6310685 >.

Haitsma, Jaap, and Ton Kalker. 2002. « A Highly Robust Audio Fingerprinting System ». In *Ismir*.

Hampapur, Arun, Ki-Ho Hyun and Ruud Bolle. 2002. « Comparison of sequence matching techniques for video copy detection ». In *Storage and Retrieval for Media Databases 2002, January 23, 2002 - January 25, 2002*. (San Jose, CA, United states). Vol. 4676, p. 194-201. SPIE. < http://dx.doi.org/10.1117/12.451091 >.

Hartung, Frank, and Martin Kutter. 1999. « Multimedia watermarking techniques ». *Proceedings of the IEEE,* vol. 87, nº 7, p. 1079-1107.

Heritier, Maguelonne, Vishwa Gupta, Langis Gagnon, Gilles Boulianne, Samuel Foucher and Patrick Cardinal. 2009. « Crim's content-based copy detection system for TRECVID ». In *TREC Video Retrieval Evaluation, TRECVID 2009, November 16, 2009 - November 17, 2009*. (Gaithersburg, MD, United states), p. National Institute of Standards and Technology (NIST). Coll. « 2009 TREC Video Retrieval Evaluation Notebook Papers »: National Institute of Standards and Technology.

Hermansky, Hynek, and Nelson Morgan. 1994. « RASTA processing of speech ». *Speech and Audio Processing, IEEE Transactions on,* vol. 2, nº 4, p. 578-589.

Indyk, Piotr, and Rajeev Motwani. 1998. « Approximate nearest neighbors: towards removing the curse of dimensionality ». In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. p. 604-613. ACM.

Jegou, H., J. Delhumeau, Yuan Jiangbo, G. Gravier and P. Gros. 2012. « BABAZ: a large scale audio search system for video copy detection ». In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2012), 25-30 March 2012*. (Piscataway, NJ, USA), p. 2369-72. Coll. « Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2012) »: IEEE. < http://dx.doi.org/10.1109/ICASSP.2012.6288391 >.

Law-To, Julien, Li Chen, Alexis Joly, Ivan Laptev, Olivier Buisson, Valerie Gouet-Brunet, Nozha Boujemaa and Fred Stentiford. 2007. « Video copy detection: A comparative study ». In *6th ACM International Conference on Image and Video Retrieval, CIVR 2007, July 9, 2007 - July 11, 2007*. (Amsterdam, Netherlands), p. 371-378. Coll. « Proceedings of the 6th ACM International Conference on Image and Video Retrieval, CIVR 2007 »: Association for Computing Machinery. < http://dx.doi.org/10.1145/1282280.1282336 >.

Le, Thi Hoang Ngan, Kim Hung Nguyen and Hoai Bac Le. 2010. « Literature survey on image watermarking tools, watermark attacks, and benchmarking tools ». In *2nd International Conferences on Advances in Multimedia, MMEDIA 2010, June 13, 2010 - June 19, 2010*. (Athens, Greece), p. 67-73. Coll. « 2nd International

Conference on Advances in Multimedia, MMEDIA 2010 »: IEEE Computer Society. < http://dx.doi.org/10.1109/MMEDIA.2010.37 >.

Lebosse, J., L. Brun and J. C. Pailles. 2007. « A robust audio fingerprint extraction algorithm ». In *Proceedings of the Fourth IASTED International Conference on Signal Processing, Pattern Recognition and Applications, 14-16 Feb. 2007*. (Anaheim, CA, USA), p. 269-74. ACTA Press.

Lezi, Wang, Dong Yuan, Bai Hongliang, Zhang Jiwei, Huang Chong and Liu Wei. 2012. « Contented-based Large Scale Web Audio Copy Detection ». In *2012 IEEE International Conference on Multimedia and Expo (ICME), 9-13 July 2012*. (Los Alamitos, CA, USA), p. 961-6. Coll. « 2012 IEEE International Conference on Multimedia and Expo (ICME) »: IEEE Computer Society. < http://dx.doi.org/10.1109/ICME.2012.17 >.

Li, Teng, Fudong Nian, Xinyu Wu, Qingwei Gao and Yixiang Lu. 2014. « Efficient video copy detection using multi-modality and dynamic path search ». *Multimedia Systems*, p. 1-11.

Li, Yuanning, Luntian Mou, Menglin Jiang, Chi Su, Xiaoyu Fang, Mengren Qian, Yonghong Tian, Yaowei Wang, Tiejun Huang and Wen Gao. 2010. « PKU-IDM@ TRECVid 2010: copy detection with visual-audio feature fusion and sequential pyramid matching ». *online] wwwnlpir. nist. gov/projects/tvpubs/tv. pubs. org. html*.

Liu, Yang, Wan-Lei Zhao, Chong-Wah Ngo, Chang-Sheng Xu and Han-Qing Lu. 2010. « Coherent bag-of audio words model for efficient large-scale video copy detection ». In *Proceedings of the ACM International Conference on Image and Video Retrieval*. p. 89-96. ACM.

Liu, Zhu, Tao Liu and Behzad Shahraray. 2009. « ATT research at TRECVID 2009 content-based copy detection ». In *TREC Video Retrieval Evaluation, TRECVID 2009, November 16, 2009 - November 17, 2009*. (Gaithersburg, MD, United states), p. National Institute of Standards and Technology (NIST). Coll. « 2009 TREC Video Retrieval Evaluation Notebook Papers »: National Institute of Standards and Technology.

Lowe, David G. 2004. « Distinctive image features from scale-invariant keypoints ». *International journal of computer vision,* vol. 60, n° 2, p. 91-110.

Lu, Jian. 2009a. « Video fingerprinting for copy identification: From research to industry applications ». In *Media Forensics and Security, January 19, 2009 - January 21, 2009*. (San Jose, CA, United states). Vol. 7254, p. The Society for Imaging Science and Technology (IS and T); The International Society for Optical Engineering (SPIE). SPIE. < http://dx.doi.org/10.1117/12.805709 >.

Lu, Jian. 2009b. « Video fingerprinting for copy identification: from research to industry applications ». In *IS&T/SPIE Electronic Imaging*. p. 725402-725402-15. International Society for Optics and Photonics.

Malekesmaeili, Mani, Mehrdad Fatourechi and Rabab K. Ward. 2009. « Video copy detection using temporally informative representative images ». In *8th International Conference on Machine Learning and Applications, ICMLA 2009, December 13, 2009 - December 15, 2009*. (Miami Beach, FL, United states), p. 69-74. Coll. « 8th International Conference on Machine Learning and Applications, ICMLA 2009 »: IEEE Computer Society. < http://dx.doi.org/10.1109/ICMLA.2009.32 >.

Malekesmaeili, Mani, and Rabab K Ward. 2014. « A local fingerprinting approach for audio copy detection ». *Signal Processing,* vol. 98, p. 308-321.

Martinez, JI, Jaime Vitola, Adriana Sanabria and César Pedraza. 2011. « Fast parallel audio fingerprinting implementation in reconfigurable hardware and GPUs ». In *Programmable Logic (SPL), 2011 VII Southern Conference on*. p. 245-250. IEEE.

Mohamed, Hisham, Hasmik Osipyan and Stephane Marchand-Maillet. 2014. « Fast large-scale multimedia indexing and searching ». In *Content-Based Multimedia Indexing (CBMI), 2014 12th International Workshop on*. p. 1-6. IEEE.

Mou, Luntian, Tiejun Huang, Yonghong Tian, Menglin Jiang and Wen Gao. 2013. « Content-based copy detection through multimodal feature representation and temporal pyramid matching ». *ACM Transactions on Multimedia Computing, Communications and Applications,* vol. 10, nº 1.

Mukai, Ryo, Takayuki Kurozumi, Kaoru Hiramatsu, Takahito Kawanishi, Hidehisa Nagano and Kunio Kashino. 2010. « NTT Communication Science Laboratories at TRECVID 2010 content-based copy detection ». In *TREC Video Retrieval Evaluation, TRECVID 2010, November 15, 2010 - November 17, 2010*. (Gaithersburg, MD, United states), p. National Institute of Standards and Technology (NIST). Coll. « 2010 TREC Video Retrieval Evaluation Notebook Papers »: National Institute of Standards and Technology.

Naphade, Milind R., Minerva M. Yeung and Boon-Lock Yeo. 1999. « Novel scheme for fast and efficent video sequence matching using compact signatures ». In *Electronic Imaging*. p. 564-572. International Society for Optics and Photonics.

Nie, Xiushan, Wenjun Zeng, Hua Yan, Jiande Sun, Zheng Liu and Qian Wang. 2014. « Structural similarity-based video fingerprinting for video copy detection ». *IET Image Processing,* vol. 8, nº 11, p. 655-661.

NIST. 2015. « Building video queries for Trecvid2008 copy detection task ». < http://www-nlpir.nist.gov/projects/tv2010/TrecVid2008CopyQueries.pdf >. Consulté le 5 Juin 2015.

NVIDIA.2015.http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf. Consulté le 1 September 2015.

Ouali, Chahid, Pierre Dumouchel and Vishwa Gupta. 2014a. « A Robust Audio Fingerprinting Method for Content-Based Copy Detection ». In *International Workshop on Content-Based Multimedia Indexing (CBMI)*. (pp. 1-6). IEEE.

Ouali, Chahid, Pierre Dumouchel and Vishwa Gupta. 2014b. « Robust Features for Content-Based Audio Copy Detection ». In *Fifteenth Annual Conference of the International Speech Communication Association*. (pp. 2395-2399).

Ouali, Chahid, Pierre Dumouchel and Vishwa Gupta. 2015a. « Content-Based Multimedia Copy Detection ». In *IEEE International Symposium on Multimedia*. (pp. 597-600). IEEE.

Ouali, Chahid, Pierre Dumouchel and Vishwa Gupta. 2015b. « Efficient Spectrogram-Based Binary Image Feature For Audio Copy Detection ». In *40th IEEE International Conference on Acoustics, Speech and Signal Processing*. (ICASSP) (pp. 1792-1796). IEEE.

Ouali, Chahid, Pierre Dumouchel and Vishwa Gupta. 2015c. « Fast Audio Fingerprinting System Using GPU and a Clustering-Based Technique ». *IEEE/ACM Transactions on Audio, Speech and Language Processing*. 24(6), 1106-1118.

Ouali, Chahid, Pierre Dumouchel and Vishwa Gupta. 2015d. « GPU Implementation of an Audio Fingerprints Similarity Search Algorithm ». In *International Workshop on Content-Based Multimedia Indexing*. (CBMI) (pp. 1-6). IEEE.

Ouali, Chahid, Pierre Dumouchel and Vishwa Gupta. 2015e. « A Spectrogram-Based Audio Fingerprinting System For Content-Based Copy Detection ». *Multimedia Tools and Applications Journal*. pp. 1-21.

Pan, Jia, and Dinesh Manocha. 2011. « Fast GPU-based locality sensitive hashing for k-nearest neighbor computation ». In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. p. 211-220. ACM.

Ramona, Mathieu, and Geoffroy Peeters. 2013. « AudioPrint: An efficient audio fingerprint system based on a novel cost-less synchronization scheme ». In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. p. 818-822. IEEE.

Riaa. 2015. « RIAA statistics ». https://www.riaa.com/physicalpiracy.php?content_selector=piracy-online-scope-of-the-problem. Consulté le June 2015.

Saracoglu, A., E. Esen, T. K. Ates, B. O. Acar, U. Zubari, E. C. Ozan, E. Ozalp, A. A. Alatan and T. Ciloglu. 2009. « Content based copy detection with coarse audio-visual fingerprints ». In *2009 Seventh International Workshop on Content-Based Multimedia Indexing (CBMI), 3-5 June 2009*. (Piscataway, NJ, USA), p. 213-18. Coll. « 2009 Seventh International Workshop on Content-Based Multimedia Indexing (CBMI) »: IEEE. < http://dx.doi.org/10.1109/CBMI.2009.12 >.

Shazam. < http://www.shazam.com/ >. Consulté le 1 Septembre 2015.

Shen, Heng Tao, Xiaofang Zhou, Zi Huang, Jie Shao and Xiangmin Zhou. 2007. « UQLIPS: a real-time near-duplicate video clip detection system ». In *Proceedings of the 33rd international conference on Very large data bases*. p. 1374-1377. VLDB Endowment.

Sivic, Josef, and Andrew Zisserman. 2003. « Video Google: A text retrieval approach to object matching in videos ». In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. p. 1470-1477. IEEE.

Soundcloud. < www.soundcloud.com >. Consulté le 1 septembre 2015.

Sui, Donghui, Li Ruan and Limin Xiao. 2014. « A Two-level Audio Fingerprint Retrieval Algorithm for Advertisement Audio ». In *Proceedings of the 12th International Conference on Advances in Mobile Computing and Multimedia*. p. 235-239. ACM.

Tanha, Maryam, Seyed Dawood Sajjadi Torshizi, Mohd Taufik Abdullah and Fazirulhisyam Hashim. 2012. « An overview of attacks against digital watermarking and their respective countermeasures ». In *2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic, CyberSec 2012, June 26, 2012 - June 28, 2012*. (Kuala Lumpur, Malaysia), p. 265-270. Coll. « Proceedings 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic, CyberSec 2012 »: IEEE Computer Society. < http://dx.doi.org/10.1109/CyberSec.2012.6246095 >.

Uchida, Yusuke, Koichi Takagi and Shigeyuki Sakazawa. 2011. « KDDI labs at TRECVID 2011: Content-based copy detection ». In *TREC Video Retrieval Evaluation, TRECVID 2011, December 5, 2011 - December 7, 2011*. (Gaithersburg, MD, United states), p. National Institute of Standards and Technology (NIST). Coll. « 2011 TREC Video Retrieval Evaluation Notebook Papers »: National Institute of Standards and Technology.

Vimeo. < www.vimeo.com >. Consulté le 1 septembre 2015.

Vobile. < http://www.vobileinc.com/ >. Consulté le 1 septembre 2015.

Voloshynovskiy, S., S. Pereira, T. Pun, J. J. Eggers and J. K. Su. 2001. « Attacks on digital watermarks: Classification, estimation-based attacks, and benchmarks ». *IEEE Communications Magazine,* vol. 39, nº 8, p. 118-125.

Wang, A. L. C. 2003. « An industrial-strength audio search algorithm ». *Proceedings of the SPIE - The International Society for Optical Engineering,* vol. 5307, nº 1, p. 582-8.

Wang, Chung-Che, Jyh-Shing Roger Jang and Wenshan Liou. 2014. « Speeding up audio fingerprinting over GPUs ». In *Audio, Language and Image Processing (ICALIP), 2014 International Conference on*. p. 5-10. IEEE.

Wu, Xiao, Alexander G. Hauptmann and Chong-Wah Ngo. 2007. « Practical elimination of near-duplicates from Web video search ». In *15th ACM International Conference on Multimedia, MM'07, September 24, 2007 - September 29, 2007*. (Augsburg, Bavaria, Germany), p. 218-227. Association for Computing Machinery. < http://dx.doi.org/10.1145/1291233.1291280 >.

Yahoo. < www.yahoo.com >. Consulté le 1 septembre 2015.

Yan, Ke, D. Hoiem and R. Sukthankar. 2005. « Computer vision for music identification ». In *Proceedings. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 20-25 June 2005*. (Los Alamitos, CA, USA) Vol. vol. 1, p. 597-604. Coll. « Proceedings. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition »: IEEE Comput. Soc.

Yeh, Mei-Chen, Chao-Yung Hsu and Chun-Shien Lu. 2010. « NTNU-Academia Sinica at TRECVID 2010 content based copy detection ». In *TREC Video Retrieval Evaluation, TRECVID 2010, November 15, 2010 - November 17, 2010*. (Gaithersburg, MD, United states), p. National Institute of Standards and Technology (NIST). Coll. « 2010 TREC Video Retrieval Evaluation Notebook Papers »: National Institute of Standards and Technology.

Younessian, Ehsan, Xavier Anguera, Tomasz Adamek, Nuria Oliver and David Marimon. 2010. « Telefonica Research at TRECVID 2010 Content-Based Copy Detection ». In *TRECVID*.

Youtube. < www.youtube.com >. Consulté le 1 septembre 2015.

Youtube. 2015. « YouTube Statistics ». < https://www.youtube.com/yt/press/statistics.html >. Consulté le 20 mars 2015.

Zhang, Xiu, Bilei Zhu, Linwei Li, Wei Li, Xiaoqiang Li, Wei Wang, Peizhong Lu and Wenqiang Zhang. 2015. « SIFT-based local spectrogram image descriptor: a novel

feature for robust music identification ». *EURASIP Journal on Audio, Speech, and Music Processing,* vol. 2015, n° 1, p. 1-15.