Content-based Video Retrieval for Pattern Matching Video Clips

Adam Jaamour

Bachelor of Science in Computer Science with Honours
The University of Bath
May 2019

Content-based Video Retrieval for Pattern Matching Video Clips

Submitted by: Adam Jaamour

COPYRIGHT

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Signed:

Abstract

Your abstract should appear here. An abstract is a short paragraph describing the aims of the project, what was achieved and what contributions it has made.

Contents

1 Introduction **1**
- 1.1 Problem Description . . . 1
- 1.2 Related Content-Based Retrieval Systems and Their Applications 3
- 1.3 Project Aim . . . 4
- 1.4 Project Structure . . . 5

2 Literature & Technology Survey **7**
- 2.1 Content-Based Retrieval . . . 7
  - 2.1.1 Video Retrieval Methods . . . 7
  - 2.1.2 Temporal Aspects of Videos . . . 9
    - Temporal Structure of a Video . . . 9
    - Challenges of Temporality . . . 10
  - 2.1.3 CBVR for Mobile Devices . . . 12
    - Query Video Quality . . . 12
    - User Experience . . . 12
- 2.2 Visual Content Extraction for Pattern Matching . . . 13
  - 2.2.1 Static Features . . . 13
    - Colour-based Features . . . 13
    - Texture-based Features . . . 16
    - Shape-based Features . . . 16
  - 2.2.2 Dynamic Features . . . 17
    - Points of Interest . . . 17
    - Object Features . . . 18
    - Motion Features . . . 18
  - 2.2.3 Learning Models for Pattern Matching . . . 19
    - Bag-of-Visual-Words . . . 20
    - Constellation Models . . . 21
    - Deep Learning Models . . . 21
- 2.3 Structural Movie Pre-Processing . . . 21
  - 2.3.1 Temporal Movie Segmentation . . . 22
    - Feature Extraction . . . 23
    - Similarity Measurements . . . 23
    - Detection . . . 24

2.3.2 Key Frame and Thumbnail Extraction 26
Key Frames 26
Thumbnails for Initial Shortlisting 28
2.3.3 Feature-Length Movie Optimisation 28
2.4 Chapter Summary 29

3 Requirements **31**
3.1 Functional Requirements 31
3.1.1 System Requirements 31
Training 31
Matching 32
Database Pre-processing 32
General Requirements 32
3.1.2 Code Design Requirements 33
3.1.3 Data Requirements 33
3.2 Non-Functional Requirements 34
3.3 Summary 34

4 Design **35**

5 Implementation **37**

6 System Testing **41**

7 Evaluation **43**

8 Discussion **45**

9 Conclusions **47**

A 13 Point Ethics Check List **53**

B Experiment Survey **55**
B.1 Title 55
B.2 What is the project about? 55
B.3 Your role in this experiment 55
B.4 Watch 56
B.5 Rank 56
B.6 Confirmation Message 57

List of Figures

1.1 Wireframe showing the basic high-level concept that the system will try to achieve. 4

2.1 Illustrations of a text-based content-retrieval system *(a)*, a content-based video retrieval system *(b)*, and the project's desired CBVR system *(c)*. 8

2.2 Temporal structure of videos, including the different terms used to describe video temporal units. Figure courtesy of A. Araujo et al. 10

2.3 Frames from the famous running scene in Forrest Gump extracted at intervals of 10 seconds. Video frames courtesy of *"Forrest Gump long run scene"* YouTube video available online: `https://youtu.be/QgnJ8GpsBG8?t=325`. 11

2.4 Example of static features, including colour-based features (with an RGB colour histograms of 256 bins for colour-based features), texture-based features (with an co-occurrence matrix showing the frequency of pairs of pixels) and shape-based features (with a shape matrix descriptor). 14

2.5 Example of a histogram counting the location of points relative to a vertical grid. Image courtesy of Bradski and Kaehler. . . . 15

2.6 If the range of the bins is too large, then the distribution is coarse (left). If the range of the bins is too small, then the distribution is not accurately represented and spikes cells appear (right). Image courtesy of Bradski and Kaehler. 16

2.7 An image of the Palace of Monaco with coloured windows representing poor features in blue (flat), edges in green, and good features in red (corners). 18

2.8 Visualisation of sparse and dense optical flow algorithms on different shots. 20

2.9 A 3D and a 2D representation of dense optical flow. Each arrow represents the direction and magnitude of a specific pixel location, which can variate in direction and length. 21

2.10 Shot boundary detection example of a video scene made up of two shots with a gradual transition between the two shots. Figure courtesy of Michael Gygli available online at: https://medium.com/gifs-ai/_ridiculously-fast-shot-boundary-detection-with-fully-conv onal-neural-networks-da9d8c73e86c 22
2.11 . Visual examples of a quick cut, a dissolve cut and a fading cut. Frames courtesy of Koprinska & Carrato (2001) "Temporal video segmentation: A survey" available online at: https://www.sciencedirect.com/science/article/pii/S0923596500000114 22
2.12 Threshold-based approach for shot boundary detection. In this example, the shot boundary is detected at A, it is missed at B, and it is misinterpreted at C as two cuts rather than a single one. 25
2.13 Example of frames to sample for low-visual content analysis. The first frame for each second (one frame every thirty seconds) is retrieved for a 30 fps 3-second video of a ball rolling from the left-hand side of the screen to the right-hand side. Video frames courtesy of *"How to Animate a Rolling Ball"* YouTube video available online: https://youtu.be/cgbLAreElNI?t=130. . . . 27

5.1 Flowchart. 39

B.1 Screenshot of the checkbox grid used to rank the database videos from most likely to match the query video to least likely. 57

List of Tables

1.1 The four types of visual search involving images and videos, classified by type of query used and by type of database being queried. 2

Acknowledgements

Add any acknowledgements here.

Chapter 1

Introduction

During the past decade, the amount of data in the world has grown at exponential rates and is currently showing no signs of slowing down. According to infographics released by IBM, already 2.7 ZB[1] of data existed in the world in 2012 [24], enough to fill up almost 40 billion 64 Gb iPhones. This number has since then risen to 8 Zb in 2015 and is expected to reach a yearly production of 35 Zb by 2020 [13] [24]. 85% of all the world data is considered to be unstructured data [7], which is mainly be made up of multimedia in the form of images and videos. An important factor in this growth is the rise of mobile devices usage and social media. Indeed, the amount of mobile-dependant users has grown at impressive rates, with now 95% of the United States citizens owning a mobile phone [15]. People tend to use their mobile devices for most day-to-day activities, with a majority of this time spent on social media services such as YouTube or Facebook. The social networks are one of the primary causes for the exponential growth of unstructured data mentioned earlier, with over 300 hours of new video content is uploaded to YouTube every minute of the day. The combination of mobile devices usage and social media data are the main contributors in today's flood of unstructured data. This project will look at creating a system combining the problem of unstructured data in the form of video data on mobile devices.

1.1 Problem Description

Content-based video retrieval is a computer vision solution to the problem of searching large databases of videos, where "content" corresponds to information from the videos such as colours or shapes that can be used to efficiently retrieve a desired video from the database. The main difficulty with content-based video retrieval lies with the reference database of videos itself. Videos in a database correspond to unstructured data, meaning only the video data[2]

[1] Zettabytes. 1 Zb = 1 trillion Gb.

[2] Video data includes the video frames and the audio.

and their metadata[3] are stored in the database. Computations involved in querying a database of videos using only the metadata available without the ability to analyse the video files would be too expensive and highly inefficient [31]. Therefore, the inspection of the video files is a necessity to find solutions to the database complication in content-based video retrieval. It is important to note that the metadata can still be used as a complement to the video files analysis to fine-tune the pattern matching. Another complication involves the large database size. Indeed, more complications arise from databases populated with a large number of videos, especially if their duration is lengthy e.g. feature-length films. Solutions to those problems hence require intelligent and efficient pattern matching algorithms to overcome the specified difficulties. This is where visual search and the variety of fields it includes such as artificial intelligence, machine learning and database management come in.

Several visual search techniques exist for querying databases of unstructured data such as video or image files. These techniques can be classified based on the type of query and on the type of database used, as shown in table 1.1. The most common forms of visual search consist in querying a database of images either with an image (I2I) or with a video (V2I), as depicted in section 1.2 where similar existing systems are mentioned. Another less frequent variant consists of querying a database of images with a video query (V2I) [5]. However, this dissertation will solely focus on querying a database of videos with a video query (V2V). Because algorithms from other variants of visual search can be relevant to V2V, existing solutions for these variants will also be explored to find ways potential techniques that could be implemented with this project.

Image Query	I2I	I2V
Video Query	V2I	V2V
	Database of Images	Database of Videos

Table 1.1: The four types of visual search involving images and videos, classified by type of query used and by type of database being queried.

To pattern match the query video to a video in the database, key elements from the query video, called *"features"*, are extracted and compared to the same features from videos in the database to find similarities. These features can include elements such as colour histograms or shapes of interest and their on-screen movements [31], as well colour percentages, colour layouts or textures [33], or other unique features such as on-screen text or audio components e.g. soundtracks, dialogues or sound effects. A wide spectrum of

[3]Examples of metadata related to a video file includes captions, file name, file type, video length, file size, etc.

existing solutions exists to efficiently retrieve videos using video queries, which are discussed in the section below.

1.2 Related Content-Based Retrieval Systems and Their Applications

Visual search technology has an endless amount of applications in fields such as education, navigation, utilities and security. Some of the systems using this technology have already found their way to commercial applications. For example, YouTube's Content ID system uses visual search, and more precisely V2V, to detect copyright infringements on user-uploaded videos by automatically comparing an uploaded video with a database of protected videos provided by their partners. If a match is found, YouTube allows these partners to take action on the uploaded video that infringed their copyrights [26]. Another example is Google Lens [2], a mobile application that recognises the input context through a mobile device's camera in order to relay information about objects of interest. For instance, if pointing the camera at a landmark, information such as historical facts or opening hours about that location will be displayed, or if pointing it at a WiFi router's label, the mobile device will automatically connect to the network [36]. Other profitable applications include A9's Amazon Flow [1] that uses deep-learning based computer vision for Amazon's search services, as well as Shazam [3], a mobile application that matches short user-recorded sounds with a piece of music.

Aside from the commercialised systems stated in the previous paragraph, many visual search applications are yet to be implemented for large-scale use. For example, a system allowing companies to detect all appearances of their logos during television broadcasts, or a system enabling students to find a section of a recorded lecture by using a lecture slide as a query [5] are possibilities that need to be explored. Content-based video retrieval academic papers have also been published in recent years. For instance, Liu et al. [28] discuss a concept similar to this project's idea consisting of a mobile visual search system allowing users to discover videos by pointing their phone at a screen. Recently, research in visual search has gained momentum, notably with the annual TRECVID[4] conference [6] hosted by the NIST[5]. This conference's goal is to host workshops that focus on information retrieval research, with an emphasis on content-based retrieval of digital videos [35]. The combination of academic research and specialised conferences will hopefully help bring more content-based retrieval solutions to real-life problems.

[4]Text REtrieval Conference Video Retrieval Evaluation

[5]National Institute of Standards and Technology

1.3 Project Aim

This project aims to find solutions to address the previously mentioned problems from Section 1.1, namely:

- the exponential rise of unstructured data in the form of media, with a focus on videos,
- the over-reliance people have on their mobile devices for basic day-to-day tasks,
- the sparse collection of existing solutions that could be combined to create more efficient systems,
- the lack of focus on videos, especially feature-length movies.

This project will attempt to translate solutions to the mentioned problems by creating a system that will combine existing solutions into a single one, with a focus on recognising a feature-length movie from a short video query recorded from a mobile device, as depicted in Figure 1.1.

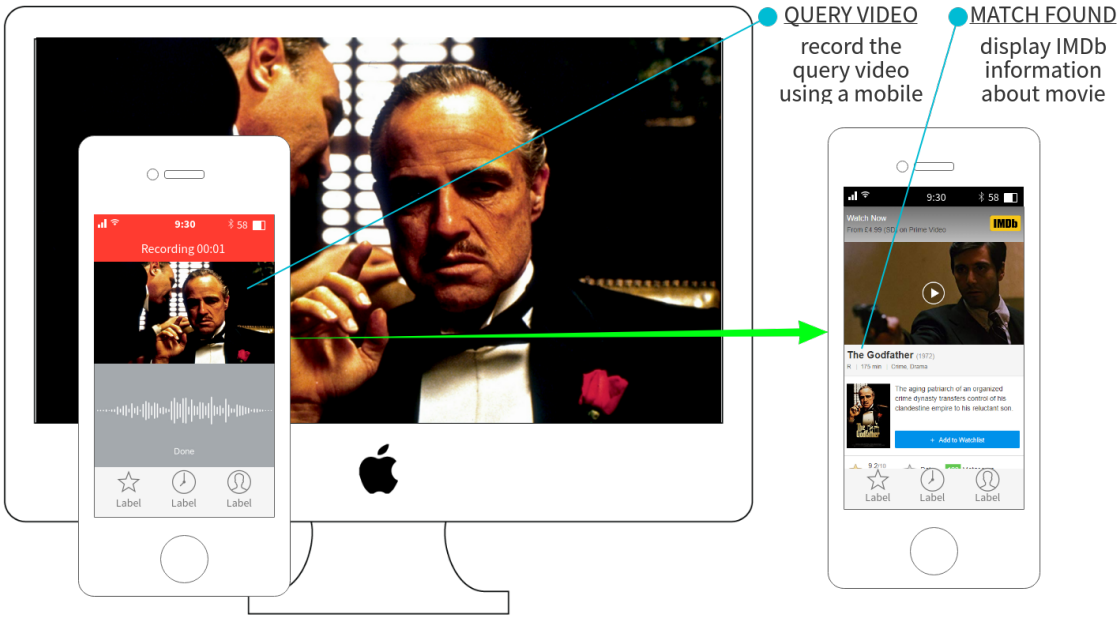


Figure 1.1: Wireframe showing the basic high-level concept that the system will try to achieve.

1.4 Project Structure

quickly describe in 1 sentence the general structure of the dissertation for each chapter

Chapter 2

Literature & Technology Survey

2.1 Content-Based Retrieval

Content-based retrieval is a type of visual search technique where large databases of either images or videos are queried to find the closest match to a query image or video. Although this project focuses on content-based video retrieval, also referred to as CBVR[1], image retrieval techniques (CBIR[2]) will be discussed as well due to their relevance in video retrieval.

This section will first review the different video retrieval methods and their evolution, starting from text-based retrieval to content-based retrieval, before addressing the various challenges that exist in video retrieval, such as the additional difficulty caused by the temporal aspect of videos compared to images, and the complications of targeting mobile devices for such a system.

2.1.1 Video Retrieval Methods

Drastic advances in video capturing technology have caused important amounts of unstructured data in the form of videos to be produced in recent years. This has lead to a high-demand to develop new efficient solutions for processing this data, with video retrieval being the answer.

Throughout the years, video retrieval has improved in parallel with the breakthroughs in video recording devices. Early video retrieval techniques used a text-based approach where the system accepted text input to search the database of videos [25], as seen in Figure 2.1.*a*. For example, the user would input the string query *"De Niro"*, which would return all movies in

[1] Content-Based Video Retrieval
[2] Content-Based Image Retrieval

which Robert De Niro starred or *"Coppola"* to find all movies directed by Francis Ford Coppola. Unique aspects of the video clip such as movie credits or sports scores were often analysed using OCR[3] technology [27]. The query text was then compared to a video file's content, such as colours, shapes, texture, luminance or objects, or to the file's metadata[4], such as the video title, author, date, content description, commentaries, captions or keywords [27] [16] [31]. However, these techniques were highly inefficient compared to content-based techniques as they often relied on manually noted annotations and textual descriptions to find similarities for matching the query video to a video in the database and did not make use of the actual visual content that describes a video.

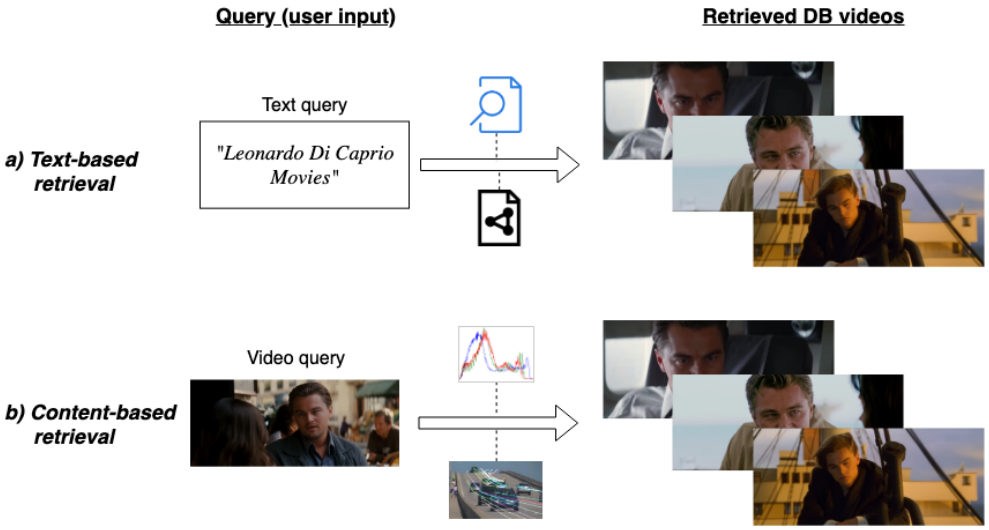


Figure 2.1: Illustrations of a text-based content-retrieval system *(a)*, a content-based video retrieval system *(b)*, and the project's desired CBVR system *(c)*.

Content-based retrieval techniques quickly replaced text-based retrieval techniques towards the end of the 20th century by making use of the visual content to compute similarities between videos [25]. Instead of accepting a text string, the system takes a video as input to extract its visual contents, as shown in Figure 2.1.*b*. According to Petković et al. [33], this visual content can be broken down into three different categories:

- *Raw data*, which corresponds to individual raw video frames and the video file's attributes such as the frame rate per seconds, the number of bits per pixel or the colour model used.

[3]Optical Character Recognition
[4]The data associated to a video file

- *Low-level visual content* consists of the visual features that describe a video. This content includes colours, shapes, textures and motion. Low-level visual content can be extracted into static features such as histograms (see Section 2.2.1) or dynamic features such as objects or motion (see Section 2.2.2) using a wide variety of existing techniques. Once the content has been extracted, it can be used to first compute similarities between videos [25] and later pattern match them.

- *Semantic content* contains the high-level concepts that are present in a video. These high-level concepts can be described as objects or events using the features. To extract semantic content from a video, a grammar of rules for objects must be provided. An example of an object rule could be "if the shape is round, the colour is orange and the object is moving, then that object is a basketball".

In comparison to raw data, low-level visual content provides the most relevant visual information that can be extracted from a video for the purpose of a CBVR system. Semantic content extraction adds an additional layer of complexity compared to low-level visual content extraction as it requires domain knowledge and user interaction [33]. Therefore, this project will focus on using low-level visual content to extract information about the video and compute the similarities between the query video and database videos. It is important to note that this project's goal differs from classic CBVR systems where a list of videos is returned (see in Figure 2.1.*b*), as it must return a specific video that matches the most the query video. To improve the pattern matching accuracy phase, raw data (e.g. audio) and metadata (e.g. captions) may be used to improve the pattern matching accuracy [31].

2.1.2 Temporal Aspects of Videos

Temporal Structure of a Video

The most important difference between content-based image retrieval and video retrieval lies within the temporal aspect of the video. Naturally, the temporal aspect of a video clip stores supplementary information about the content, including dynamic low-level visual content e.g. an object's motion, and semantic content e.g. actions and events. According to A. Araujo et al. [5], a video's temporal structure can be subdivided into three units, as shown in Figure 2.2:

- *Frames* correspond to the smallest temporal unit of a video file. A single segment of a video is referred to as a frame. Frames are also used to describe the frame rate (the frequency at which consecutive stills appear on a screen every second) e.g. "24 fps" corresponds to a video made up of 24 stills per second.

- *Shots* are grouped sequences of visually similar frames. They are usually described in seconds.

- *Scenes* are collections of shots which are related based on the action and objects present in the shot, thus giving them a semantic aspect. The length of a scene is generally calculated in minutes rather than seconds.

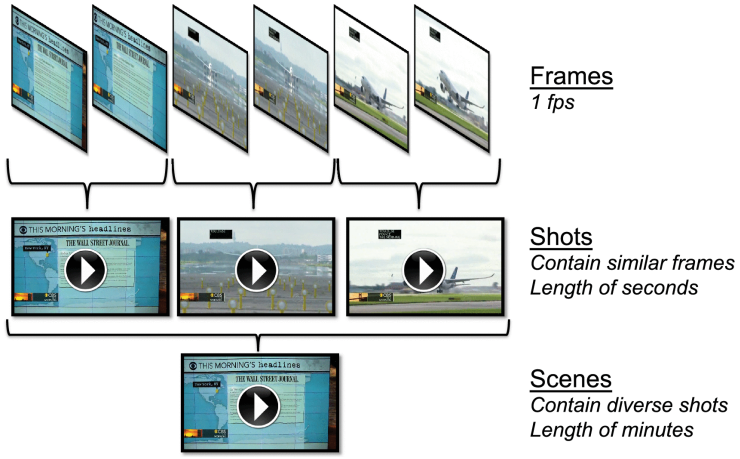


Figure 2.2: Temporal structure of videos, including the different terms used to describe video temporal units. Figure courtesy of A. Araujo et al.

Because this project will explore possible solutions to create a CBVR system targeting databases of feature-length movies, a fourth video temporal structure category relevant to this project can be added to Araujo et al.'s initial list:

- *Movies* can be described as a large group of scenes that are used to tell a story. Movie durations commonly range from one to three hours.

Challenges of Temporality

Multiple challenges arise when dealing with CBVR systems in contrast to CBIR systems as the videos' temporal aspect adds a new dimension of complexity when extracting visual information. While the low-level visual content describing images remains mostly the same for videos, some new information that did not exist in images can be extracted, such motion. As mentioned previously, videos are made up of frames, which make up shots when a combination of similar frames are played in succession. This means that videos carry information about motion, such as the trajectory of objects. This introduces unique challenges to the algorithms used to extract motion.

Because videos are made up of numerous stills, usually around 24 frames per second [9], two consecutive frames are near-identical. The pixels describing an object in one frame will remain the same in the next frame, except for the edge pixels perpendicular to the motion's trajectory [8].

Figure 2.3 shows six frames from Forrest Gump's famous running shot, which lasts 44 seconds, making up a total of 1056 frames. Each frame in the figure was captured with 10-second intervals, meaning 240 frames separate each still. In the first three frames, the group of people running in the background barely moves in the space of 20 seconds. The pixels that describe the group in the shot remain mostly unchanged for all of the frames between the 3 samples, equivalent to 480 frames, with a few additional pixels describing the group as it advances towards the camera. The same can be said about the red cap in the last two frames. Most of the pixels making up the cap in the fifth frame remain the same in the sixth frame. This example perfectly betrays the reason why analysing a video frame by frame would be extremely inefficient when it comes to CBVR. Due to the similarities between consecutive frames, these should be aggregated [5] to describe a shot by using a selection of frames, such as taking the six frames in Figure 2.3 to describe the entire 44 seconds of video, rather than keeping the original 1056 frames to describe it.

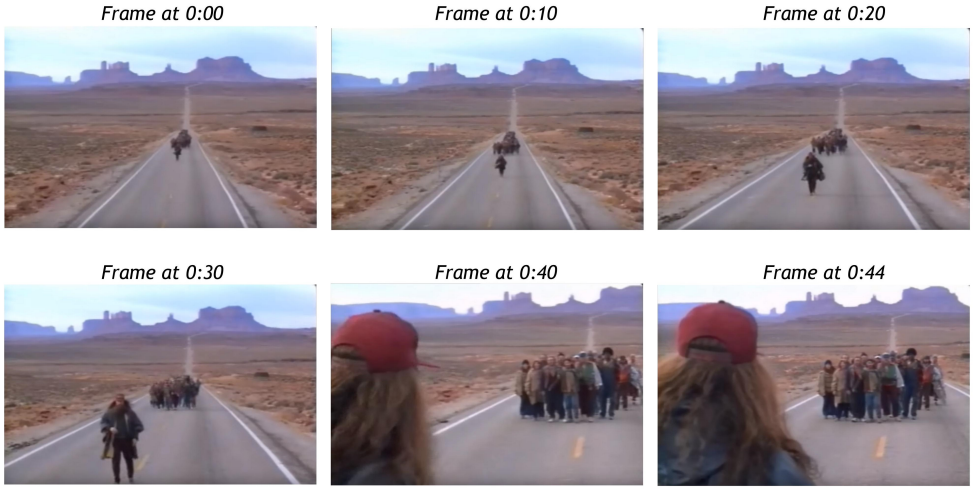


Figure 2.3: Frames from the famous running scene in Forrest Gump extracted at intervals of 10 seconds. Video frames courtesy of *"Forrest Gump long run scene"* YouTube video available online: `https://youtu.be/QgnJ8GpsBG8?t=325`.

2.1.3 CBVR for Mobile Devices

The aforementioned project aim is for the system to work on mobile devices for two reasons. The first reason is to allow users to directly use their mobile phone to record the query video by pointing their camera to a screen displaying a movie, which will, in turn, tell them which movie is being played (see the wireframe in Figure 1.1). Additional information retrieved from IMDb[5], such as cast, crew, ratings, runtime and synopsis could also be displayed. The second reason is the popularity of mobile devices, which may be due to the improvements made on mobile phones' processing power, allowing more tasks to be carried out through this medium. However, such a system on a mobile device causes many problems regarding the query video recording method and the computational power available on mobile devices.

Query Video Quality

Large visual differences are caused between the query clip and the actual clip stored in the database due to the capture conditions [28] [37] such as:

- Undesired camera movements due to unstable recording e.g. unstable recording, hand shaking.
- Low-quality recording due to poor user recording e.g. scaling and rotation, and due to the environmental conditions e.g. lighting, reflections, blurring.
- Video noise because of the camera sensor.
- Decoding artefacts caused by various file compression.

These low-quality conditions add difficulty to the pattern matching phase where the similarities between the query video and database videos have to be computed. Indeed, if the query video is very different from the actual video, then the noisy elements of the video query must be filtered out. For example, if the recorded video is shaky, then this shaking motion has to be pruned before analysing the recorded clip's motion. However, processing power must be used from the actual visual content extraction and pattern matching phases to be used for video noise filtering.

User Experience

According to Liu et al. [28], the majority of mobile device users expect a polished product with quick video query and instant or progressive results, meaning that the searching algorithms must be efficient. However, one of the

[5]Internet Movie Database

downsides of mobile devices is the computation power constraints. Despite the improvements in mobile processors, desktop devices still remain more powerful than mobile ones. A solution that Liu et al. suggest is to retrieve the low-level visual content locally on the mobile device and send the query to a server where the pattern matching will take place [28]. This allows heavy computations to be off-loaded from the mobile device. Once a match is found, the result is returned to the user on his mobile device. A downside to this approach is the new constraint on network bandwidth rather than computational power.

2.2 Visual Content Extraction for Pattern Matching

Extracting the visual content from a video allows this content to be used to describe videos and compute similarities between them. This visual content is extracted from the aforementioned low-level visual content (see Section 2.1.1) [33] and stored in the form features, also referred to as visual descriptors. The term "features" is very broad and can be used to describe many different visual aspects in an image or in a video, ranging from colours, shapes and textures to points, edges, objects and motion.

These features can be divided into two categories: static features and dynamic features [33]. This section will first survey examples of static visual descriptors and methods to extract them from videos, and will then focus on examples and methods of extracting dynamic visual descriptors from videos.

2.2.1 Static Features

Methods to extract static features operate on stills, which can correspond to individual video frames, thumbnails or key frames (see Section 2.3). This means that traditional image techniques can be applied to those stills [21]. They are organised in three different categories: colour-based features, texture-based features and shape-based features, which can visualised in Figure 2.4.

Colour-based Features

The main colour-based features model are colour histograms. In general, a histogram consists of counts of some underlying data that is organised into predetermined bins to a statistical representation of the distribution of that data. Figure 2.5 depicts an example of a histogram where a collection of points is organised into specific pre-defined bins based on their location relative to a vertical grid.

In the case of a colour histogram, the underlying data that the histogram is trying to represent is the distribution of the colour pixels throughout an image

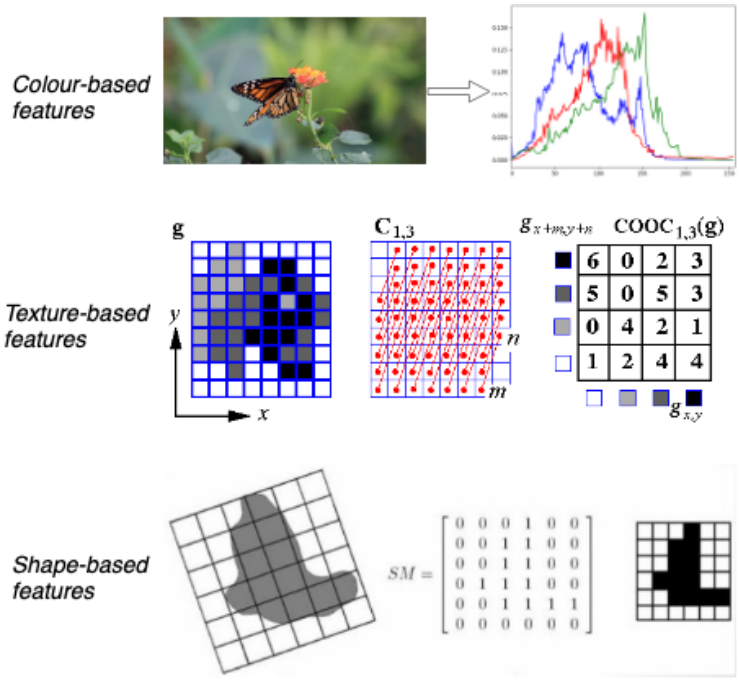


Figure 2.4: Example of static features, including colour-based features (with an RGB colour histograms of 256 bins for colour-based features), texture-based features (with an co-occurrence matrix showing the frequency of pairs of pixels) and shape-based features (with a shape matrix descriptor).

or a video frame. Different kinds of colour histograms exist as they depend on the chosen colour space, which includes RGB[6], HSV[7], HSL[8] or YPbPr colour spaces to name a few. These may vary based on the applications of the colour histograms.

Typically, for an RGB colour histogram, 256 bins are used to accurately represent all the possible values that the pixels can take (ranging from 0 to 255) for each of the three RGB channels, which are then plotted as three individual graphs. Choosing the right range for the histogram's bins is crucial to represent the distribution efficiently. If the range of pixels that defined the bins is wider, meaning there are less overall bins, then the histogram's distribution would be too coarse-grained and the general structure of the histogram would be lost, as pointed out by the left half of Figure 2.6. On the other hand, if the range of the bins is too narrow, meaning there are more overall bins, then the histogram's distribution would not be represented accurately and there would

[6]Red Green Blue
[7]Hue Saturation Value
[8]Hue Saturation Light

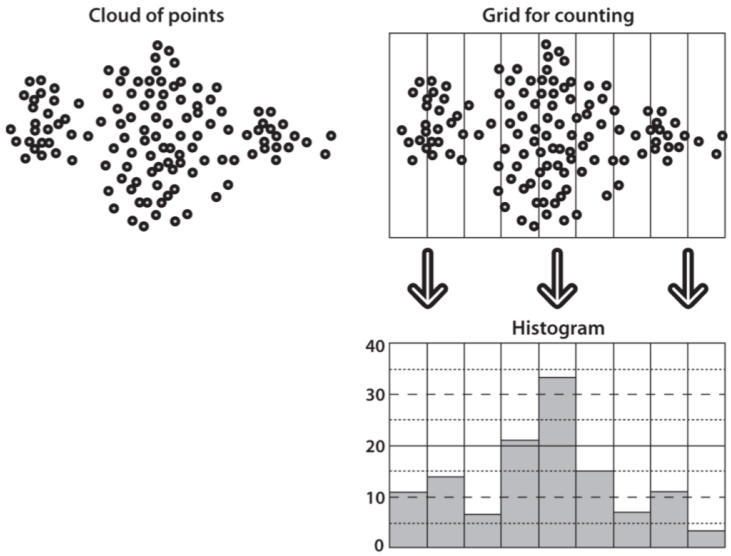


Figure 2.5: Example of a histogram counting the location of points relative to a vertical grid. Image courtesy of Bradski and Kaehler.

be many spiky cells, as betrayed in the right half of Figure 2.6 [8].

One of the inaccuracies with colour histograms lies within the scope of the distribution. If the histogram represents the global distribution of all the pixels in the still, then two images might have very similar histograms [33]. For example, a histogram containing 60% white pixels and 40% blue pixels could either describe both a blue sky with white clouds or a snowy landscape with a blue sky. Despite both histograms being good colour-based features, the actual result is still poor when it will be used for matching the histograms. A solution consists in segmenting the still into multiple local images, and extract the local colour-based features for each segment. For example, the still could be partitioned into a 5x5 grid, and a colour histogram could then be computed for each grid [38]. This would enable colour-based features to represent specific regions of the still rather than globally describing an image. However, the same problem mentioned earlier could occur if the still is segmented into too many regions, causing the overall histogram to be coarse.

Other types of colour-based features can be extracted from images and videos such as colour moments, colour correlograms [22] and Gaussian models. However, colour-based features have their limitations as they cannot describe textures and shapes, rendering them inefficient in certain applications [21].

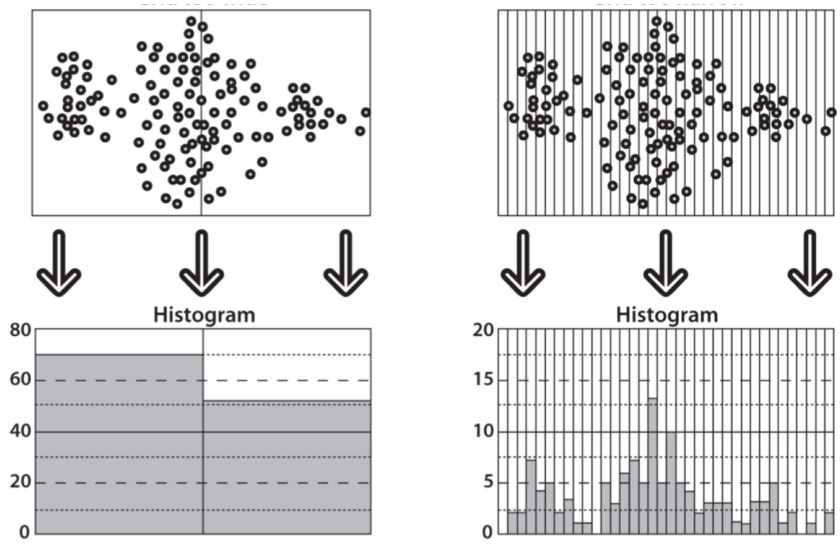


Figure 2.6: If the range of the bins is too large, then the distribution is coarse (left). If the range of the bins is too small, then the distribution is not accurately represented and spikes cells appear (right). Image courtesy of Bradski and Kaehler.

Texture-based Features

Texture-based features are often used in parallel with colour-based features. The aforesaid problem where two different objects might share a similar histogram can be solved by using texture-based features to differentiate them e.g. green tree leaves and green grass. These can be discerned by using a variety of features such as Tamura features, which extract information including coarseness, contrast, and directionality of the objects [4].

Shape-based Features

Shape-based features are used to describe the overall shape of objects present in the image. The most common approach consists in computing an EHD[9], which consists in detecting the edges present in the image (see Section edges) and then plot their spatial distribution in a histogram by counting the number of pixels for each edge [18]. The same image segmentation technique can be used to localise the EHD. These shape-based features have many applications but are harder to extract and require more computing power than colour-based features and texture-based features.

[9]Edge Histogram Descriptor

2.2.2 Dynamic Features

In contrast to static features, which can be extracted from individual video frames, dynamic features require the continuity between consecutive frames to extract relevant visual descriptors, making use of the temporal aspect of the video mentioned in section 2.1.2). These features can be divided into two subcategories: object features and motion features. However, before reviewing the techniques used to extract these features, it is important to specify what defines a good visual feature.

Points of Interest

Figure 2.7 shows an image with coloured windows used to make the difference between poor and good potential features that could be used for object and motion features:

- *Flat surfaces* are portrayed in blue in Figure 2.7. These blue windows are spread over large areas of the image, meaning it is difficult to find their specific location. Moving the blue window along the image in any direction will result in the same visual content being represented in the window. Therefore, these flat regions are the worst structures as they do not contain any useful information.

- *Edges* are characterised in green in Figure 2.7. These are more informative than flat surfaces as they can be more accurately localised, but pinpointing an exact location is still hard as the patch can be moved in the direction parallel to the edge. Moving the green window along the edge will again result in the same visual content being represented in the window. Edges are efficient to detect object boundaries, but not for tracking specific points.

- *Corner* are characterised in red in Figure 2.7. These are the most descriptive points as they are often unique and can be precisely located in an image. Moving the red window in any direction will cause it to look different. Corners are therefore the ideal candidate for features used in object matching and tracking.

Once expressive and unique descriptors like corners are detected, they can be used to extract object features and motion features, and to compute similarities between the query video and the videos in the database.

Many different algorithms exist to find robust features and interest points. For example, Sobel operators can be used to detect edges, and Shi and Tomasi algorithms and Harris operators can be used to detect corners [8]. Combinations of both can be used, by detecting edges to facilitate corner detection.

Figure 2.7: An image of the Palace of Monaco with coloured windows representing poor features in blue (flat), edges in green, and good features in red (corners).

Object Features

Object features correspond to objects that are detected using the colour, texture and size of image regions. Some of the most common objects usually detected in videos are faces, as many CBVR systems use them to compute similarities between videos [34]. However, extracting object features is time-consuming and expensive in terms of required processing power, which is why CBVR algorithms either focus on detecting specific sets of objects rather than general objects that may be present in a scene, or on static features.

Motion Features

Motion features are a unique characteristic of videos that are absent from images. All of the above-mentioned features can be extracted from images, apart from motion features, which are unique to videos. These can originate from two sources: either background camera movement or foreground object movement. Motion features can therefore be classified into two categories: camera-based motion features and object-based motion features. *Camera-based motion features* include movement such as camera zooms, pannings and tiltings. *Object-based motion features* are more interesting than the former since they can describe key objects in the shot, which can be further classified into [21]:

- *Statistics-based* motion features are extracted to model the local and global distributions of motion points in the video. For example, a causal Gibbs model can be used to represent the spatiotemporal distribution of local motion measurements in a shot once the trajectory-based motion features have been used to prune undesired camera motions [14]. This type of motion features is very cheap to extract but lacks depth as they cannot represent object actions and object relationships accurately.

- *Trajectory-based* motion features are extracted by representing object trajectories in a shot. A long list of algorithms exist to obtain these features such as the, but this section will focus on differentiating sparse and dense techniques with one of the most popular methods: optical flow. Optical flow can be defined as the relative motion between the visual content in a shot and the camera [8]:

  In *sparse optical flow*, only a few pixels from the frames in a shot are used. Features describing objects, such as corners, are used to track an object's motion across the shot. However, these only give information about where certain parts of the object are going, which are then used to estimate where the overall object is moving towards. Sparse optical flows. For example, Figure 2.8 indicates the overall motion of a car by tracking only a few pixels from it. On the other hand, *dense optical flow* calculates the optical flow for every pixel in the frame. At each pixel of the frame, the direction and magnitude of that specific location are represented by an arrow, which can variate in direction and length (see Figure 2.9).

  Object-based motion features have many applications outside of CBVR systems. One of these applications is calculating the optical flow of a shot to prune camera movements caused by unstable recording e.g. hand shaking movement [37]. Compared to statistics-based features, trajectory-based features can be used to describe object actions, but they rely on multiple challenging tasks to function efficiently such as correct object tracking and automatic trajectory recording. Finally, object-based motion features can also be classified in a third category: objects' relationship-based motion features. However, these are not described in this literature review as they make use of the objects' symbolic representations, which is not relevant to this topic.

2.2.3 Learning Models for Pattern Matching

This section retraces the evolution of pattern matching models, ranging from Bags-of-Visual-Words and Constellation Models to Deep Learning models such as Neural Networks.

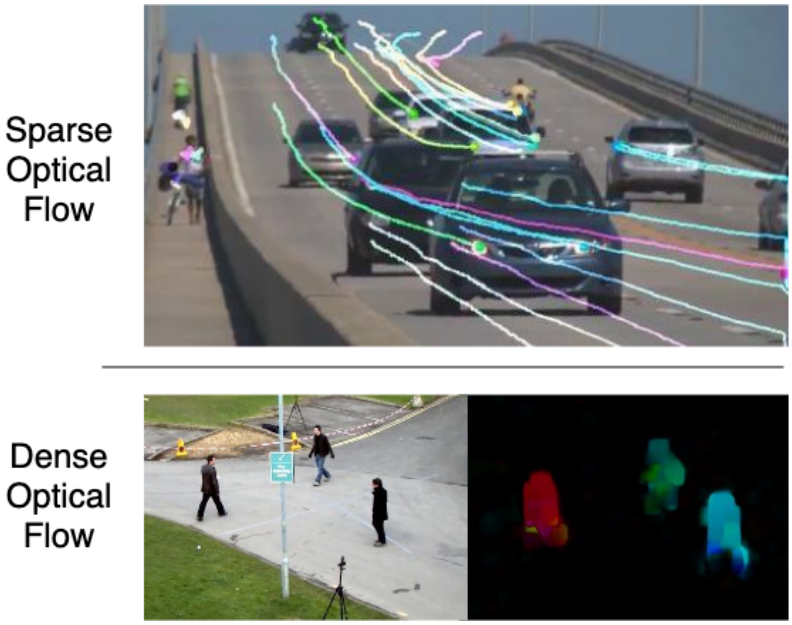


Figure 2.8: Visualisation of sparse and dense optical flow algorithms on different shots.

Bag-of-Visual-Words

The idea of BoVW[10] originates from document analysis, where the frequency words appeared in the text depended on the document type, and the document type could predict the word frequency.

Visual words are created by extracting distinctive features such as SIFT, SURF or HoG to train videos from a training set. These features are then clustered using classic models such as Mean-shift or GMMs. The visual words correspond to the middle of each cluster.

Histograms can be used by inputting a collection of visual words and a labelled training set of videos. For each video in the dataset, features are extracted and incremented in the related histogram. Once the training is accomplished, a new histogram can be generated for a query video (in a CBVR system) which will then be compared to the previously generated histograms for each dataset video. Example: H. Wang's implementation of BoW histogram [37]. Fisher Vectors are an improvement on BoW [5].

[10] Bag-of-Visual-Words

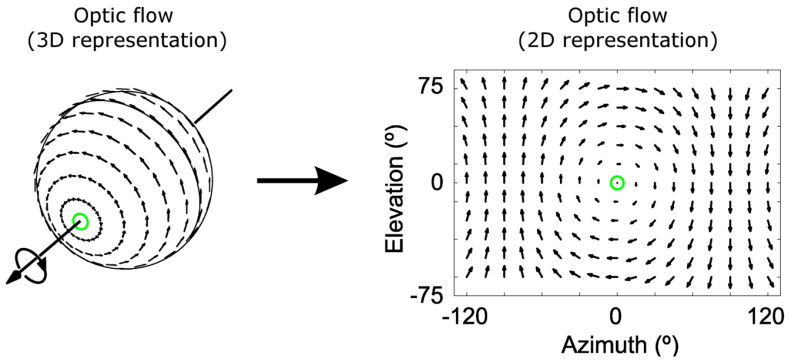


Figure 2.9: A 3D and a 2D representation of dense optical flow. Each arrow represents the direction and magnitude of a specific pixel location, which can variate in direction and length.

Constellation Models

Constellation models are mainly used for detection. Features are organised in space such as a graph model where nodes are labelled with appearance and a Gaussian distribution is present on each node location. There is one model per class, s in a CBVR system, one model per video. The model is moved over image (could be video keyframes), which fires responses at each location. The biggest responses that are larger than a threshold locate the model.

Deep Learning Models

- Classical Neural Networks
- Convolutional Neural Networks
- State of the art: AlexNet, VGG, Inception

2.3 Structural Movie Pre-Processing

The database of videos can be pre-processed to optimise the visual content extraction and pattern matching phases. As stated in Section 2.1.2 on the temporal aspects of videos, movies can be defined as a logically ordered collection of scenes, which contain multiple shots, all made up of individual frames. Shots are therefore the logical fundamental unit to organise and segment a video into [21]. Pre-processing the database of movies by segmenting it into a list of shots and representing each shot with a single key frame is a profitable solution that will exponentially improve a CBVR system's efficiency. However, a limit must be set on the amount of data that is segmented to balance the efficiency and the accuracy of the CBVR system.

2.3.1 Temporal Movie Segmentation

The frames that make up a shot usually show strong content correlation. This means that features extracted from one frame will be extremely similar in another frame from the same shot. Detecting shot boundaries would allow the movie to be segmented and organised by shots, as depicted in Figure 2.10. These boundaries are defined by the type of transition between two different shots, which can either be defined as a quick cut when the transition is direct, or as gradual when it is a dissolve or a fade in/out transition that runs over multiple frames [39], as shown in Figure 2.11.

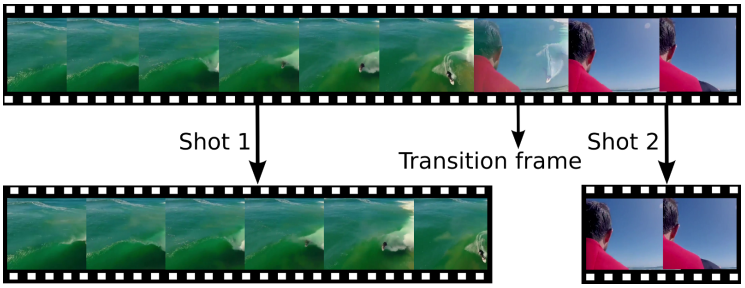


Figure 2.10: Shot boundary detection example of a video scene made up of two shots with a gradual transition between the two shots. Figure courtesy of Michael Gygli available online at: `https://medium.com/gifs-ai/_ridiculously-fast-shot-boundary-detection-with-fully-convoluti_onal-neural-networks-da9d8c73e86c`

Figure 2.11: . Visual examples of a quick cut, a dissolve cut and a fading cut. Frames courtesy of Koprinska & Carrato (2001) "Temporal video segmentation: A survey" available online at: `https://www.sciencedirect.com/science/article/pii/S0923596500000114`

The goal of SBD[11] is to detect the discontinuity between consecutive frames. Also know as temporal video segmentation, this process requires three steps:

1. feature extraction
2. similarity measurements
3. shot boundary detection

Feature Extraction

The first step in video segmentation consists in extracting features. These features include all the diverse types mentioned in Section 2.2, ranging from static features such as colour-based features (e.g. colour histograms) [20] to dynamic features such as corner points and SIFT[12]. Although colour histograms are simple to compute and work well with most shots as long as there is at most minor camera movement, they are inefficient when the shots contain major camera movements [21]. For example, if in the shot the camera moves from the inside a house towards the outside by going through the window, the frames retrieved from the beginning of the shot will be extremely different to frames from the end of the shot.

Dynamic features are therefore more efficient for shot boundary detection than histograms due to their robustness. On the one hand, edge features are more vigorous than histograms when dealing with major camera movement and can handle changes in luminosity. On the other hand, corner and motion features can additionally handle camera motion and the impact of objects in the shot such rapid motion e.g. people walking in front of the camera. However, these dynamic features are more complicated to extract and do not always outperform simple features like colour histograms. Due to their simplicity, colour histograms remain the most common feature extraction technique for shot boundary detection [39].

Similarity Measurements

The second step in video segmentation is to use these extracted features to compute the similarities between frames. Many metrics exist to compute the similarities between two the extracted features. The most basic method consists in computing the Euclidean distance or the absolute distance $g(n, n+k)$ between a pair of frames [23, p.476], as shown in Equation 2.1, where n and

[11] Shot Boundary Detection
[12] Scale Invariant Feature Transform

$n+k$ represent the two frames being compared, and $I_n(x, y)$ the intensity level of frame n at pixels locations (x, y):

$$g(n, n+k) = \sum_{x,y} |I_n(x, y) - I_{n+k}(x, y)| \tag{2.1}$$

More advanced distance metrics can be used based on the type of feature that was extracted. For instance, intersection and chi-square distances can be used to compute the distance between the histograms of two frames [10, p.197]. During the 2006 TRECVID conference, *Hoi et al.*, used grey scale histograms for their extracted features and calculated the colour differences between frames using the EMD[13]. EMD is used to calculate the distance between two probability distributions, which are represented by the two histograms that represent each frame in [20, p.2]'s case. These similarities can be measured using two techniques: pair-wise similarities and window similarities.

The first technique rests on using pairs of frames to compute the similarities between them. The similarities between two consecutive frames I_1 and I_2 are compared, before comparing the pair of frames I_2 and I_3, and so on and so forth until frames I_{n-1} and I_n are compared. This straightforward technique precisely detects rough changes between the visual content in a shot (e.g. quick cuts between two shots) but is more sensitive to intensity changes caused by noise and disturbances such as camera motion and objects, leading to a high level of wrong detections [23].

The second technique is the window-based similarity measure, that measures the similarities between multiple frames specified within a range [11]. This helps counter the effects of noise and disturbances captured by the pair-based approach but is more expensive to compute, therefore less used [21].

Detection

The final step in segmenting a video is the detection of the frames to cut, which is achieved by using the measured similarities between the frames. Two methods exist for this step: a threshold-based approach and a statistical learning-based approach.

The *threshold-based approach* compares the measured similarities between a pair of frames with a threshold. Whenever the similarity measure is smaller than the threshold, a shot boundary is detected and the shot can be cut. Figure 2.12 betrays an example of a threshold-based approach for different types of transitions, with a correct detection (A), a missed detection (B) and a misinterpreted detection (C).

[13] Earth Mover's Distance

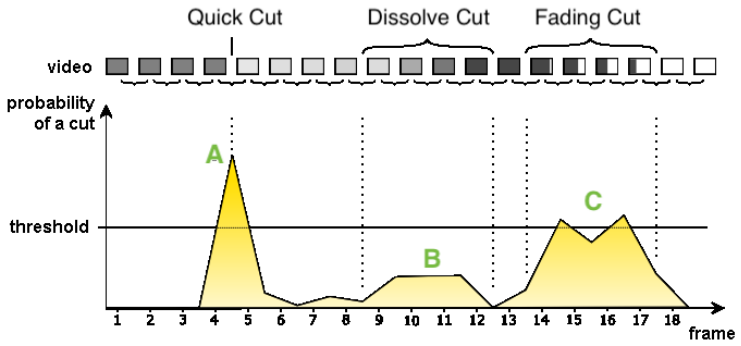


Figure 2.12: Threshold-based approach for shot boundary detection. In this example, the shot boundary is detected at A, it is missed at B, and it is misinterpreted at C as two cuts rather than a single one.

Either global thresholds, adaptive thresholds, or combinations of both can be used [11] [21]:

- *Global thresholds* are set empirically and predefined for the entire video. They are therefore inefficient when trying to detect local variations as these are not integrated into the initial global threshold calculations.

- *Adaptive thresholds* are local thresholds that are continuously calculated for windows of frames. As the window slides across the video, the threshold is re-estimated accordingly to account for local variations. These are more accurate than global thresholds but are harder to estimate and require some knowledge about the video to set variables such as the window's size.

- *Combination of global and adaptive thresholds* are adaptive thresholds that take into account the values of global thresholds when estimating the frames within the window. Different thresholds can be determined for different types of transitions such as cut and dissolve transitions. However, these add increased complexity to the threshold estimation.

The *statistical learning-based approach* is a two-class classification task where each frame is classified either as a "shot change" or as a "no shot change" based on the extracted features and the similarity measurements between those features. In more complex systems, additional classes can be added by adding a classification of quick and gradual transitions for example. Two types of classifiers can be used for this approach:

- *Supervised learning-based classifiers*, such as the Support Vector Machine [10] and Adaboost [40, p.617]. These supervised learning approaches

have multiple advantages compared to threshold-based approaches as they do not require thresholds to be set and many different types of features can be combined in the feature extraction step to increase the classifier's accuracy. However, this approach requires more expertise as the classifier relies on a well-trained data set to remain accurate [21].

- *Unsupervised learning-based classifiers* are divided into frame similarity-based and frame-based algorithms. *Frame similarity-based classifiers* groups the similarity measurements between a pair of frames into two clusters. The first cluster holds the similarity measurements with low values, which correspond to the "shot change" class, while the second cluster holds the similarity measurements with high values, which correspond to the "no shot change" class. K-Means and Fuzzy K-Means clustering algorithms can be used for frame similarity-based classifiers [29]. Alternatively, *frame-based classifiers* consider each shot as a cluster of frames with similar features. Clustering ensembles can be used to classify frames into their respective shots [12]. This approach is advantageous over supervised learning-based classifiers as no prior training is required for, but the logical temporal order of the shots in scenes is not preserved.

All the aforementioned techniques in this section that are used in the three steps required in the process of segmenting a video can be applied to feature-length movies but require some improvements to work efficiently with these large videos. For instance, [17] suggests considering the semantic aspects of movies, since people relate to the story, such as a marking dialogue, when they remember a movie.

2.3.2 Key Frame and Thumbnail Extraction

Frames from the same shot usually describe the same visual content, meaning there is a lot of redundancy amongst the frames that make up a shot. It is therefore more efficient to deal with a single frame that represents the entire shot or the entire video rather than dealing with an entire video.

Key Frames

Once shot boundaries have been detected and the video has been divided into a series of shots, a single frame, named the "key frame", can be selected to represent the entire shot. [19] suggests only considering key frames in videos to operate on. These key frames would be used for all the sections mentioned in this Literature Survey, including the visual content extraction phases, the pattern matching phases, and database videos pre-processing phase. A wide variety of approaches exist to extract a key frame from a video.

The most simple approach is *Sequential Comparison Between Frames* where frames are sequentially compared until a frame very different to the previous key frame is found, at which point that frame is set as a new key frame [19]. Frames are compared using the features mentioned in Section 2.3.1 e.g. histograms, and similarities are calculated using the same distance functions mentioned in Section 2.3.1 e.g. chi-square distance [21]. More advanced approaches can be used:

- Global Comparison Between Frames
- Reference Frame
- Clustering
- Curve Simplification
- Objects/Events

To illustrate the advantage of using key frames, a 3-seconds long shot recorded at 30 fps[14] of a ball rolling on the ground can be used (90 frames in total). Analysing all the frames individually as stills would obviously be highly inefficient. However, selecting key frames to work on, as depicted in Figure 2.13 where a single frame is chosen for each second, would mean that 3 key frames can be used for feature extraction and pattern matching instead of using all of the 90 frames that make up the video.

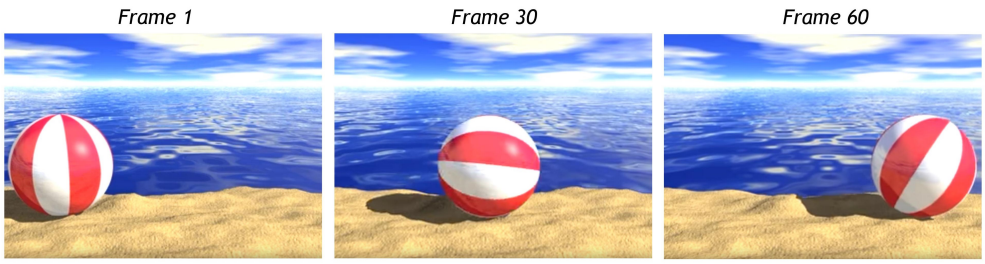


Figure 2.13: Example of frames to sample for low-visual content analysis. The first frame for each second (one frame every thirty seconds) is retrieved for a 30 fps 3-second video of a ball rolling from the left-hand side of the screen to the right-hand side. Video frames courtesy of *"How to Animate a Rolling Ball"* YouTube video available online: `https://youtu.be/cgbLAreElNI?t=130`.

[14]Frames Per Second

Thumbnails for Initial Shortlisting

Thumbnails can be generated for each video in a database, which are stored as additional data along with the original video file [30]. In a CBVR system, the thumbnails for the query video and for the database videos would be generated using the same algorithm in order to create similar outputs. This technique can be used in parallel to [5], who states that an initial shortlist of potentially matching videos can be generated before the main pattern matching phase. This shortlist can be created by computing the similarities between the query video's thumbnail and the thumbnails of the database videos.

Many advantages can be gained from this small initial step which could have an important impact on the system's overall speed and efficiency performance. Indeed, database videos that share no similarities to the query video will not be considered at all during the main pattern matching phase e.g. if the query video corresponds to a colourful sunset, then database videos of cloudy environments will be immediately filtered out as it is unlikely that they will match with the query video in the main pattern matching phase. This step diminishes the number of videos to compare to the query.

This initial step of generating thumbnails for the query video and each video in the database is extremely speedy as it only uses a single frame that describes the entire video. Therefore the entire process will not be noticeably slowed down. However, this concept could only be applied to relatively small databases where the videos contain a single scene with multiple similar shots. Indeed, shots can be characterised by a single still as they are made up of frames with similar visual content, whereas scenes can be made up of shots that change a lot and describe different visual content. Using thumbnails to shortlist a database of feature-length movies would greatly diminish the accuracy of the system due to the high number of scenes that make up a movie that cannot be resumed to a single thumbnail. The database videos' thumbnails will have already been generated during the database's pre-processing phase, which only occurs a single time. A possible solution could be to create a thumbnail for each shot in a scene and store each thumbnail in a list. For example, if a scene contains six different shots, then six thumbnails will be generated to describe that scene.

2.3.3 Feature-Length Movie Optimisation

For quicker pattern matching and feature extraction, 16 pixels separate two feature points, allowing for rough pattern matching rather than exact pattern matching. This could be used when using feature-length movies as database videos to pattern match [30].

Fisher Vectors can be used to generate a compact signature for videos, which will then be used for computing similarities between the query video's FV and the database videos' FVs [5].

2.4 Chapter Summary

Include a chapter summary here.

Chapter 3

Requirements

This chapter establishes the requirements of the system. These requirements were devised in response to the problems mentioned in Section 1.1 of Chapter 1, along with the information provided by the Literature and Technology Survey in Chapter 2.

The following requirements are prioritised using the words *must*, *should* and *may*; where *must* indicates a crucial requirement to the system, *should* an optimal requirement and *may* a beneficial requirement.

3.1 Functional Requirements

3.1.1 System Requirements

Training

These requirements concern the training phase of the system, where features are extracted from each video in the database and stored in files for the next phase.

1. **The system must be trained a single time only**.
   The system **must** re-train the classifier whenever a change is carried out in the database e.g. a video has been added or a video has been removed.

2. **The system must store the training data in files.**
   Files can be saved either in "*.txt*" file format or in "*.dat*" file format.

3. **The files containing the training data must be able to be quickly loaded into memory during the matching phase.**
   To minimise additional time spent on I/O operations, the files should be loaded quickly using efficient functions.

4. **The system should extract static colour-based features into histograms.**
   It should use either RGB histograms, grey scale histograms, HSV histograms, or a combination of multiple of the aforementioned histograms.

5. The system's interface **should** specify that it is processing the database videos by displaying either a spinner or iteratively printing text specifying which video is being processed.

Matching

These requirements concern the matching phases, or testing phase, of the system, where the same features extracted in the training phases are extracted from the query video and compared to stored features from each database video by calculating a similarity score.

1. **The system must accept a pre-recorded video as input.**
   This pre-recorded video will serve as the query video.

2. **The system must crop the recorded video to select a "region of interest".**
   It **must** ignore the pixels outside of the selected area of the clip.

3. **The system must display the results once the classification has been completed.**
   The correct result **must** be emphasised. The results can be displayed in tables if all the scores are shown.

4. **The system should stabilise the query video before pattern matching it against the videos in the database.**
   The stabilised video will diminish the effects of camera movement and improve the accuracy of the histogram matching algorithm.

Database Pre-processing

todo

General Requirements

1. The MVP[1] **must** be a CLI[2].

2. The interface **must** accept command line inputs.

3. The system **must** be able to run on a machine equipped with a terminal.

[1] Minimum Viable Product
[2] Command Line Interface

4. The system **should** be able to run on Windows, Mac and Linux distributions.
5. The system **should** be able to run on any desktop device equipped with a Python environment running with version 3.7 or higher.
6. The system **should** run smoothly.
7. The system **must** include a "README" file with instructions on how to use the system.
8. The interface **should** be able to either directly record a video or to use a pre-recorded video as input.
9. The interface **may** be a graphical interface e.g. a web page or a native desktop application.

3.1.2 Code Design Requirements

1. The system code **must** be written in Python version 3.7 or higher.
2. The system code **must** use the OpenCV python library.
3. The system code **must** be source controlled using Git.
4. The system code **must** be backed up on GitHub.
5. The system code's GitHub repository **must** be private during the development phases.
6. The system **must** be covered with unit tests.
7. The python code **should** follow PEP8 coding guidelines [32].
8. The system **should** be covered with integration tests.
9. The system **may** be covered with acceptance tests.

3.1.3 Data Requirements

1. The database videos **must** not violate any copyright laws.
2. The video data length **must** range from 10 to 15 seconds.
3. There **must** be a minimum of 20 unique videos in the database.
4. The video database **must** easily be implemented with Python.
5. The videos in the database **should** vary in style e.g. include movie extracts, animated pictures, cartoons, mangas, black and white movies, etc.

3.2 Non-Functional Requirements

1. The interface **should** be easy to read, using large fonts, with an emphasis on text used to discern the main results from .

2. The system **may** not crash or exit unexpectedly with no error message.

3.3 Summary

This chapter has summarised the requirements needed to design and implement a successful system based on the problems formulated in Chapter 1 and the Literature and Technology Survey in Chapter 2. Now that the requirements have been established, different potential solutions will be explored in the next chapter, including the benefits and disadvantages of each method, before one solution can be chosen and implemented.

Chapter 4

Design

- Explore possible options for different sections of the system, such as:
  - types of features to extract (why static colour features and not object/motion features) - histograms are very popular with videos, calculations are easier, implementation is easier, results are as efficient
  - types of learning models (histogram matching, BoW-approacg VS Neural Network)
  - programming language of choice:
    - python VS MATLAB vs other languages
    - rich array of libraries: OpenCV, NumPy, SciPy, MatplotLib
    - ease of installing third-party libraries with PIP
    - powerful IDEs (PyCharm)
    - ease of generating testing suites
    - familiarity with
  - interface choice (why CLI VS GUI?): time constraints and no users, goal of this project is to research efficient results, not create a commercial product
  - types of videos in databases (why short videos VS movies, cartoons/stop-motion pictures)
- Finish by mentioning the final solution details.
- high-level diagram of system excluding detail (all DB videos in system, single query video in system, matching video output)

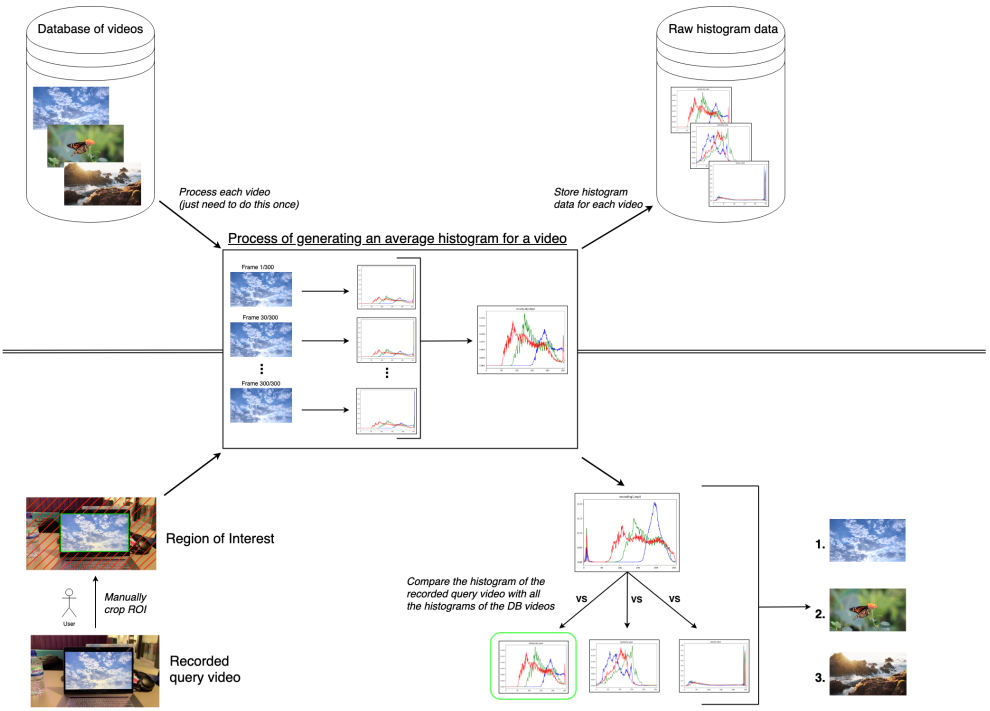
Chapter 5

Implementation

Follow low-level implementation detail of the different stages of the pipeline:

- systems architecture overview (general architecture of a histogram classifier, the different steps to make it work)
- Offline colour-based feature extraction phase:
  - generating gray scale histograms
  - generating RGB histograms
  - generating HSV histograms
  - histogram normalisation
  - saving data to human-readable .txt files
- Online retrieval phase:
  - query video, identical features extracted, video stabilised
  - distance measurements to find nearest neighbour between query histograms and each database video histogram:
    - correlation
    - intersection
    - chi-square
    - alternative chi-square
    - bhattacharyya
    - kullback leibler divergence
    - wassertein distance (Earth's Mover Distance)
    - energy distance
  - combining results from all 3 methods with weight between different histogram methods to give more importance to HSV, GRB and less importance to grey scale: 1-5-10

- Database **pre-processing** phase:
  - shot boundary detection for long videos, allowing the entire movie to be represented with a few thousand frames
  - current: one frame every second for short videos
- interface/output related:
  - opencv box drawing to select Region of Interest
  - matplot lib histogram plotting (with an option to show each generated histogram or to hide them)
  - tables to show scores for each type of histogram and each distance
  - exporting results to CSV for further analysis
  - spinners to show calculations are being done when training the system or stabilising videos
  - argument parsing
  - final output to clearly show if result if right/wrong, including a frame of the original query video, a frame of the match found, along with the runtime and tha accuracy
- include code snippets for individual sections



flowchart.png

Figure 5.1: Flowchart.

Chapter 6

System Testing

- Unit Testing
  - grey scale / RGB / HSV histogram generation
  - histogram saving to file
  - histogram normalisation
  - ROI selection
  - individual histogram matching, test each metric
- Integration Testing
  - test the entire pipeline (with a few videos as the database, make sure the correct match is found)
- Include some code snippets
- Include test suite results

Chapter 7

Evaluation

- details about the dataset
  - test with 50 database videos
  - query: test with different videos at different angles with +/- hand movement
- show quantitative values with:
  - number of True Positives
  - number of False Positives
  - general accuracy
  - runtime
- analyse results of RGB, grey scale and HSV histograms with the 8 different histogram matching methods. Mention which work best?
- online classification experiment
  - Google Survey experiment results
  - Compare experiment results with the algorithm results
  - use a plot to visualise experiment results
  - link to appendix (Ethics Checklist Appendix A, Experiment Script Appendix B, and Raw Results)

Chapter 8

Discussion

- state what could be improved with time
  - region-based histograms, separate frame in 5 regions and generate a histogram for each region rather than having a global histogram to describe the frame (https://www.pyimagesearch.com/2014/12/01/complete-guide-building-image-search-engine-python-opencv/)
  - improve ROI selection to take 4 points rather than 2 (will increase accuracy for query videos that are at an angle, not filming the screen from a straight point of view)
  - automatic ROI detection (detect screen edges with edge and corner detector algorithms)
  - more features to complement the colour-based features (e.g. motion and shape features) and improve the accuracy
  - make use of sound to further improve accuracy
  - GUI e.g. Tkinter application or simple HTML webpage
- mention that with improvements, could have worked with small database of movies rather than short videos
- limitations of working with large database of movies:
  - too much data to process, could perhaps be pre-processed in advance
  - copyright issues of having all these movies

Chapter 9

Conclusions

This is the chapter in which you review the major achievements in the light of your original objectives, critique the process, critique your own learning and identify possible future work.

Bibliography

[1] A9: Visual search. `https://www.a9.com/what-we-do/visual-search.html`. [Online] Accessed: 2018-10-28.

[2] Google lens. `https://lens.google.com/`. [Online] Accessed: 2018-10-28.

[3] Shazam. `https://www.shazam.com/gb/company`. [Online] Accessed: 2018-10-28.

[4] Arnon Amir, Marco Berg, Shih-Fu Chang, Winston Hsu, Giridharan Iyengar, Ching-Yung Lin, Milind Naphade, Apostol Natsev, Chalapathy Neti, Harriet Nock, et al. Ibm research trecvid-2003 video retrieval system. *NIST TRECVID-2003*, 7(8):36, 2003.

[5] A. Araujo and B. Girod. Large-scale video retrieval using image queries. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(6):1406–1420, June 2018.

[6] George Awad, Asad Butt, Keith Curtis, Yooyoung Lee, Jonathan Fiscus, Afzal Godil, David Joy, Andrew Delgado, Alan F. Smeaton, Yvette Graham, Wessel Kraaij, Georges Quénot, Joao Magalhaes, David Semedo, and Saverio Blasi. Trecvid 2018: Benchmarking video activity detection, video captioning and matching, video storytelling linking and video search. In *Proceedings of TRECVID 2018*. NIST, USA, 2018.

[7] Robert Blumberg and Shaku Atre. The problem with unstructured data. *Dm Review*, 13(42-49):62, 2003.

[8] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*, pages 493–636. " O'Reilly Media, Inc.", 2008.

[9] Kevin Brownlow. Silent films: What was the right speed? *Sight and Sound*, Summer:164–167, 1980.

[10] G Camara-Chavez, F Precioso, M Cord, S Phillip-Foliguet, and A de A Araujo. Shot boundary detection by a hierarchical supervised approach.

In *2007 14th International Workshop on Systems, Signals and Image Processing and 6th EURASIP Conference focused on Speech and Image Processing, Multimedia Communications and Services*, pages 197–200. IEEE, 2007.

[11] Zuzana Cernekova, Ioannis Pitas, and Christophoros Nikou. Information theory-based shot cut/fade detection and video summarization. *IEEE Transactions on circuits and systems for video technology*, 16(1):82–91, 2006.

[12] Yuchou Chang, Dah-Jye Lee, Yi Hong, and James Archibald. Unsupervised video shot detection using clustering ensemble with a color global scale-invariant feature transform descriptor. *EURASIP Journal on Image and Video Processing*, 2008(1):860743, 2007.

[13] Maria Deutscher. Big data is the new natural resource: New ibm tool powers decision making. `https://siliconangle.com/2012/04/03/big-data-is-the-new-natural-resource-new-ibm-tool-powers-decision-making/`, April 2012. [Online] Accessed: 2018-10-28.

[14] R. Fablet, P. Bouthemy, and P. Perez. Nonparametric motion characterization using causal probabilistic models for video indexing and retrieval. *IEEE Transactions on Image Processing*, 11(4):393–407, April 2002.

[15] Jason Fanning, Sean P Mullen, and Edward McAuley. Increasing physical activity with mobile devices: a meta-analysis. *Journal of medical Internet research*, 14(6), 2012.

[16] Bailan Feng, Juan Cao, Xiuguo Bao, Lei Bao, Yongdong Zhang, Shouxun Lin, and Xiaochun Yun. Graph-based multi-space semantic correlation propagation for video retrieval. *The Visual Computer*, 27(1):21–34, Jan 2011.

[17] A. Hanjalic, R. L. Lagendijk, and J. Biemond. Automated high-level movie segmentation for advanced video-retrieval systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(4):580–588, June 1999.

[18] Alex Hauptmann, Robert V Baron, Ming-yu Chen, Mike Christel, Pinar Duygulu, Chang Huang, Rong Jin, W-H Lin, T Ng, and Neema Moraveji. Informedia at trecvid 2003: Analyzing and searching broadcast news video. Technical report, 2004.

[19] Yong Seok Heo, Soochahn Lee, and Ho Yub Jung. Consistent color and detail transfer from multiple source images for video and images. *The Visual Computer*, 32(10):1273–1289, Oct 2016.

[20] Steven CH Hoi, Lawson LS Wong, and Albert Lyu. Chinese university of hongkong at trecvid 2006: Shot boundary detection and video search. In *TRECVid 2006 Workshop*, pages 76–86. Gaithersburg, Maryland, NIST, 2006.

[21] Weiming Hu, Nianhua Xie, Li Li, Xianglin Zeng, and Stephen Maybank. A survey on visual content-based video indexing and retrieval. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(6):797–819, 2011.

[22] Jing Huang, S Ravi Kumar, Mandar Mitra, Wei-Jing Zhu, and Ramin Zabih. Image indexing using color correlograms. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:762, 1997.

[23] Nitin J Janwe and Kishor K Bhoyar. Video shot boundary detection based on jnd color histogram. In *2013 IEEE Second International Conference on Image Information Processing (ICIIP-2013)*, pages 476–480. IEEE, 2013.

[24] Douglas Karr. Big data brings marketing big numbers. `https://martech.zone/ibm-big-data-marketing/`, May 2012. [Online] Accessed: 2018-10-28.

[25] Y. Lai and C. Yang. Video object retrieval by trajectory and appearance. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(6):1026–1037, June 2015.

[26] Krüger Lars. Improving rights management at youtube. `https://youtube-creators.googleblog.com/2012/04/improving-rights-management-at-youtube.html`, 2012.

[27] Huiping Li and D. Doermann. Video indexing and retrieval based on recognized text. In *IEEE Workshop on Multimedia Signal Processing.*, pages 245–248, Dec 2002.

[28] W. Liu, T. Mei, and Y. Zhang. Instant mobile video search with layered audio-video indexing and progressive transmission. *IEEE Transactions on Multimedia*, 16(8):2242–2255, Dec 2014.

[29] Chi-Chun Lo and Shuenn-Jyi Wang. Video segmentation using a histogram-based fuzzy c-means clustering algorithm. *Computer Standards & Interfaces*, 23(5):429–438, 2001.

[30] Makoto Okabe, Yoshinori Dobashi, and Ken Anjyo. Animating pictures of water scenes using video retrieval. *The Visual Computer*, 34(3):347–358, Mar 2018.

[31] B. V. Patel and B. B. Meshram. Content based video retrieval. *The International Journal of Multimedia & Its Applications*, 4(5):77–98, October 2012.

[32] PEP8. Pep 8: Style guide for python code. `https://www.python.org/dev/peps/pep-0008/`, 2001. [Online] Accessed: 2019-03-27.

[33] M. Petković. Content-based video retrieval. In *Extending DataBase Technology PhD Workshop*, 2000.

[34] Josef Sivic, Mark Everingham, and Andrew Zisserman. Person spotting: video shot retrieval for face sets. In *International conference on image and video retrieval*, pages 226–236. Springer, 2005.

[35] Alan F. Smeaton, Paul Over, and Wessel Kraaij. Evaluation campaigns and trecvid. In *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, pages 321–330, New York, NY, USA, 2006. ACM Press.

[36] Antonio Villas-Boas. Google's new lens product uses your phone's camera to do clever tricks, like connecting your phone to a wifi network. `http://uk.businessinsider.com/googles-lens-feature-can-connect-your-phone-to-wifi-using-your-camera-2017-5?op=1&r=US&IR=T`, 2017.

[37] Heng Wang, Dan Oneata, Jakob Verbeek, and Cordelia Schmid. A robust and efficient video representation for action recognition. *International Journal of Computer Vision*, 119(3):219–238, Sep 2016.

[38] Rong Yan and Alexander G Hauptmann. A review of text and image retrieval approaches for broadcast news video. *Information Retrieval*, 10(4-5):445–484, 2007.

[39] Jinhui Yuan, Huiyi Wang, Lan Xiao, Wujie Zheng, Jianmin Li, Fuzong Lin, and Bo Zhang. A formal study of shot boundary detection. *IEEE transactions on circuits and systems for video technology*, 17(2):168–186, 2007.

[40] Zhi-Cheng Zhao and An-Ni Cai. Shot boundary detection algorithm in compressed domain based on adaboost and fuzzy theory. In *International Conference on Natural Computation*, pages 617–626. Springer, 2006.

Appendix A

13 Point Ethics Check List

This document describes the 13 issues that need to be considered carefully before students or staff involve other people ("participants") for the collection of information as part of their project or research.

1. **Have you prepared a briefing script for volunteers**?
   All participants will be introduced with a briefing script explaining what the project consists in, what their role in the experiment is, and how their data will be used.

2. **Will the participants be using any non-standard hardware?**
   No. Participants can conduct the experiment from their personal computers or mobile devices.

3. **Is there any intentional deception of the participants?**
   No deception will occur during this experiment, all the details of the experiment will be provided in the debriefing script.

4. **How will participants voluntarily give consent?**
   The participants will give consent when they submit their Google Form survey, which is specified in the briefing script.

5. **Will the participants be exposed to any risks greater than those encountered in their normal work life?**
   Participants will take part in the experiment from their normal environments as they will be receive a link to the online survey in their personal inbox, which they can use to access the survey online from their personal computers and mobile devices.

6. **Are you offering any incentive to the participants?**
   No incentive will be offered to participants.

7. **Are any of your participants under the age of 16?**
   All participants are above the age of 18.

8. **Do any of your participants have an impairment that will limit their understanding or communication?**
   None of the participants have any known impairment which may affect the experiment's results.

9. **Are you in a position of authority or influence over any of your participants?**
   The participants will complete the survey in their own free time using their own hardware with no presence of authority present during the experiment.

10. **Will the participants be informed that they could withdraw at any time?**
    All the participants will be informed that they can withdraw from the experiment at any time in the briefing script.

11. **Will the participants be informed of your contact details?** My contacts details and my supervisor's details will be shared with all participants.

12. **Will participants be de-briefed?**
    All participants will be debriefed through the introductory briefing script.

13. **Will the data collected from the participants be stored in an anonymous form?**
    The data collected from the online survey will be downloaded and stored locally in a CSV file anonymously, and will be used anonymously in the final report, as informed in the briefing script.

Appendix B

Experiment Survey

The survey used for the experiment. The survey was created using Google Forms and is available here: `https://forms.gle/HyQAGyks2Tnj7wgc9`

B.1 Title

Content-Based Video Retrieval Experiment

B.2 What is the project about?

This dissertation project aims to create a system similar to Shazam for movies. For background information, Shazam is a mobile application that allows users to match a recording to a piece of music.

The goal of this project is to explore how such a system could be developed to work with movies, where a user would record a screen with his phone, which will pattern match the movie to a database of movies and accurately return the movie title.

B.3 Your role in this experiment

In this experiment, you play the role of the algorithm that matches a user recording to one of the videos in the database.

You will be shown 6 videos from the database, followed by 1 user-recorded video.

Your goal is to mentally compare the recorded video to each video in the database to be able to rank each database video from most likely to match

the recorded video to least likely.

When making your decision, take into account aspects of the video such as colour distribution, luminosity, textures, shapes, objects and motion, as a video matching algorithm would in real life.

Note: The data retrieved from this experiment will be compared to the results the of the project's matching algorithm. It will remain anonymous in locally-stored hard copies and in the final report. By submitting this form, you give consent for your data to be evaluated and used in the report. You may withdraw from the experiment at any time.

B.4 Watch

Watch the 6 database videos that will train your "knowledge".

Database Videos: `https://www.youtube.com/watch?v=BvukbK-sX9A`.

Now watch the recorded query video. Your goal is now to compare this video to the 6 previous videos you watched and to rank them based on their visual similarities.

Query Video: `https://www.youtube.com/watch?v=4JPo0-aSzNE`.

B.5 Rank

Which database video does the recorded query match the most?

See Figure B.1

Which video aspects did you consider the most when ranking them?

Placeholder for long answer from participant.

What do you think is the most important aspect of a video that a matching algorithm should analyse when pattern matching videos?

Placeholder for short answer from participant.

	Video A (snowy winter)	Video B (Earth from space)	Video C (shuttle landing)	Video D (cloudy sky)	Video E (sunset)	Video F (beach)
Most likely to match (#1)	☐	☐	☐	☐	☐	☐
2nd most likely to match (#2)	☐	☐	☐	☐	☐	☐
3nd most likely to match (#1)	☐	☐	☐	☐	☐	☐
4th most likely to match (#4)	☐	☐	☐	☐	☐	☐
5th most likely to match (#5)	☐	☐	☐	☐	☐	☐
Least likely to match (#6)	☐	☐	☐	☐	☐	☐

Figure B.1: Screenshot of the checkbox grid used to rank the database videos from most likely to match the query video to least likely.

B.6 Confirmation Message

(The message that is displayed to users once they have finished the experiment and submitted the Google Form).

> Your response has been recorded.
>
> Thank you taking part in this experiment!
>
> Here are my algorithm's results for comparison:
>
> 1. cloudy-sky (video D)
> 2. winter (video A)
> 3. shuttle-landing (video C)
> 4. sunset (video E)
> 5. beach (video F)
> 6. earth (video B)
>
> Contact details:
>
> - Adam Jaamour: *aj645@bath.ac.uk*
> - Dr. Yong-Liang Yang (project supervisor): *Y.Yang2@bath.ac.uk*