Content-based video summarization in Object Maps

by Manuel Martos Asensio

directed by Horst Eidenberger and Xavier Giró-i-Nieto

Technische Universität Wien (TU Wien)
Universitat Politècnica de Catalunya (UPC)
2012-2013
Contents

List of Figures ................................................................. v
List of Tables ................................................................. viii
Agraïments ........................................................................ ix
Acknowledgments ............................................................. x
Abstract .............................................................................. xi
1 Introduction ......................................................................... 1
  1.1 Focus of the thesis .......................................................... 2
  1.2 Motivation ...................................................................... 3
  1.3 Applications ................................................................... 4
  1.4 Outline of the thesis ....................................................... 4
2 Related work ......................................................................... 6
  2.1 Definitions ...................................................................... 6
  2.2 Shot segmentation .......................................................... 8
  2.3 Content selection ........................................................... 13
  2.4 Object detection ............................................................. 19
3 Requirements ........................................................................ 30
  3.1 Scope of the thesis .......................................................... 30
  3.2 Requirements analysis .................................................... 31
  3.3 Overview and priorities ................................................... 33
4 Design and Implementation .................................................. 35
  4.1 Overview ...................................................................... 35
  4.2 Shot segmentation .......................................................... 37
  4.3 Face detection ............................................................... 39
List of Figures

Fig. 1 Video summary example of relevant faces .............................................................. 2
Fig. 2 Shot boundary detection example ............................................................................... 8
Fig. 3 Shot detection example using Hausdorff distance method ...................................... 9
Fig. 4 Shot boundary detection using UCSD pixel regions difference ............................... 10
Fig. 5 False shot detection useful for the project. Frontal and side views .......................... 11
Fig. 6 Shot boundary detection using NN ............................................................................ 13
Fig. 7 Framework of user attention model [7] .................................................................... 15
Fig. 8 Mosaic representation. Top: Hand-chosen keyframes. Bottom: Mosaic representation without foreground occlusions [23] ................................................................. 17
Fig. 9 Visual story lines example [27] ................................................................................ 18
Fig. 10 Deformable part model detection [41] ................................................................... 20
Fig. 11 Haar-like features used in OpenCV ......................................................................... 21
Fig. 12 First two features selected in Viola-Jones algorithm .............................................. 22
Fig. 13 Summed Area Table example .................................................................................. 22
Fig. 14 Cascade classifier for face detection ......................................................................... 23
Fig. 15 Discrete Gaussian second derivative box filters ....................................................... 25
Fig. 16 Haar-wavelet in x and y directions .......................................................................... 25
Fig. 17 SURF descriptor performance in different image intensity patterns ....................... 26
Fig. 18 LBP code creation example ...................................................................................... 26
Fig. 19 LBP invariant to monotonic grayscale transformations .......................................... 26
Fig. 20 Star model of a person category [41] ....................................................................... 28
Fig. 21 Matching process for Deformable parts-based models approach [41] ................. 29
Fig. 22 Proposed system architecture .................................................................................. 35
Fig. 23 Frontal (yellow) and profile (blue) detections over extracted keyframe ............... 39
Fig. 24 Face detection stages architecture .......................................................................... 40
Fig. 25 Results for frontal and profile face detections ........................................... 41
Fig. 26 Removed detections with size filtering ....................................................... 42
Fig. 27 Removed profile detection overlapping with frontal one ............................. 43
Fig. 28 Face detection output ................................................................................ 43
Fig. 29 Profile detection example .......................................................................... 43
Fig. 30 Pre-processing facial images for face features extraction ........................... 45
Fig. 31 Face clustering algorithm block diagram ................................................... 47
Fig. 32 First iteration of face clustering block ....................................................... 48
Fig. 33 Second iteration of face clustering block .................................................... 48
Fig. 34 Final iteration of face clustering block ....................................................... 48
Fig. 35 Car object detection examples .................................................................... 50
Fig. 36 SURF training images .............................................................................. 51
Fig. 37 SURF descriptors extraction ..................................................................... 51
Fig. 38 SURF matching examples ......................................................................... 52
Fig. 39 Root filter of car model7 ........................................................................... 53
Fig. 40 Different view car detection using libpabod .............................................. 54
Fig. 41 Left: Source image. Right: Partition image ............................................... 55
Fig. 42 Left: Detected faces with bounding boxes. Right: Positive oval marker in red, negative marker in blue ................................................................. 55
Fig. 43 Segmented face ......................................................................................... 56
Fig. 44 Face maps with largest shot background representation ............................ 56
Fig. 45 Face maps with largest face background representation ........................... 57
Fig. 46 Example of tile-based map with void regions ........................................... 58
Fig. 47 Tile-based composition examples ............................................................... 59
Fig. 48 GAT user interface .................................................................................... 62
Fig. 49 Web-based evaluation survey .................................................................... 68
Fig. 50 Global rating of the summaries .................................................................. 69
Fig. 51 Individual rating of the summaries ............................................................. 69
Fig. 52 Object map summary for trailer 1, The Intouchables .................................. 70
Fig. 53 Uniform sampling summary for trailer 1, The Intouchables ....................... 70
Fig. 54 Object map summary for trailer 7, The Fast and the Furious ..................... 70
Fig. 55 Global recognition rate ............................................................................ 71
Fig. 56 Individual recognition rate ........................................................................ 71
Fig. 57 Uniformly sampled summary of trailer 4, Dark Shadows ......................... 72
Fig. 58 Uniformly sampled summary of trailer 9, Resident Evil 5: Retribution ....... 72
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Requirements overview and priorities</td>
<td>34</td>
</tr>
<tr>
<td>2</td>
<td>Overview of requirements and architecture blocks</td>
<td>36</td>
</tr>
<tr>
<td>3</td>
<td>Video items used in the user study</td>
<td>66</td>
</tr>
<tr>
<td>4</td>
<td>Object map summary setup</td>
<td>73</td>
</tr>
</tbody>
</table>
Agraïments

Voldria aprofitar aquesta oportunitat per expressar la meva gratitud als meus directors de projecte, Prof. Horst Eidenberger i Prof. Xavier Giró-i-Nieto, per la seva paciència, el seu suport incessant i estímul Durant la Tesi. Els consells inspiradors del Prof. Eidenberger Durant la meva estada a Viena i la crítica perspicaç del Prof. Giró-i-Nieto durant els últims anys en la meva beca de recerca a la UPC són extremadament essencials i valuoses per aquesta Tesi. Aquest treball no s’hauria completat sense el seu esforç.

M’agradaria també agrair al department de Processament d’Imatge de la UPC que m’ha ajudat a solucionar problems amb la integració de diferents tecnologies utilitzades en el projecte.

També agraeixo els valiosos suggeriments i comentaris que la Clara m’ha donat. La meva estada a Viena ha estat molt més agradable amb ella al meu costat.

Finalment, però no per això menys important, agraïments especials a la meva família que m’han donat el millor suport i estímul per venir a Viena i acabar els meus estudis d’Enginyeria Superior.
Acknowledgments

I would like to take this opportunity to express my gratitude to my supervisors, Prof. Horst Eidenberger and Prof. Xavier Giró-i-Nieto, for their patient guidance, ceaseless support and encouragement during my Thesis. The inspiring advice from Prof. Eidenberger during my stay in Vienna and the insightful criticism from Prof. Giró-i-Nieto over recent years in my research grant at UPC are extremely essential and valuable in this Thesis. This work could not be completed without their effort.

I would also like to show my gratitude to the UPC Image Processing Department that helped me in solving integration problems of the different technologies used in the Thesis.

I am also grateful for the valuable suggestions and comments that Clara has given to me. My stay in Vienna has become much more pleasant with her by my side.

Last but not least, my special thanks must go to my family who has given me the greatest support and encouragement, so that I can come to Vienna and finish my master studies.
Abstract

The amount of digital video content available on the web is constantly increasing. Its handling requires efficient technologies: text search on large databases provides users a great amount of videos; the content results are accessible by a description. Users need a fast and visual way to access relevant video content effectively. Quick visualization of content using static image summarization is a sophisticated problem. However, it is worth it because it may solve video navigation problems. Users can very rapidly get an idea of the video with no need to browse through it with a sliding bar as normally done.

In this work a system for automatic video summarization is developed. It creates an object map, the segments of which are extracted from an input video. It allows enhancing video browsing and large video databases management generating a visual index so that the user can rapidly grasp the most relevant content. Finally, accessing them with a simple action requires several technologies that define a complex information processing.

Firstly, shot boundary detection algorithms are required to reduce time redundancy of the video. Secondly, different relevant objects are extracted from each keyframe (faces, cars, etc.). We also describe a workflow to train detection models using multiple open source solutions. Furthermore, faces are a particular and very relevant semantic class. For this reason, we use clustering methods in order to recognize them in an unsupervised recognition process. The image composition of all selected objects and faces is the final stage of the architecture. Composition is defined as the combination of distinct parts to form a whole, therefore, objects have to be rendered in the map in a visually attractive manner.

To validate our approach and assess end-user satisfaction, we conducted a user study in which we compare requirements collected by analyzing related literature. We analyze redundancy and informativeness as well as pleasantness.

The results show that our approach effectively creates an image representation for videos and is able to summarize customizable content in an attractive way.
1

Introduction

The volume of video content is growing every day. The manipulation, interaction and management of large video collections are far from other types of media such as text or images; one of the main reasons is the temporal nature of video. Text searches can be done in many ways, for instance, search commands on single words with very specific metadata. On the other hand, images have thumbnail representations for rapid image browsing. Furthermore, new portable devices, such as smart phones or tablets, along with social networks and user-generated content sites greatly increase the accessibility and production of videos. Normally, video search results are accessible by textual metadata but it is not always the best way to summarize a video. Shared content requires efficient retrieval technologies to access the content properly in a fast and intuitive way.

This thesis addresses the problem of video content summarization using relevant objects, analyzing the video and helping users to understand a video content item in a fast and visual way. Automatic video summarization aims at improving video browsing and temporal search of digital multimedia content supporting users in navigation of large videos archives.

Our approach for automatic video summarization into an object map is based on content analysis. Object mapping is the process of taking data from one form of representation (video) to another (image). The research aims at complementing the capabilities of summaries over other media summaries, such as text summaries, using relevant content extraction.
1.1 Focus of the thesis

We design an automatic system from existing algorithms that can create efficient image representations of video content items to help users detect important objects as well as providing a quick navigation through them.

Selecting the main content for the summary is performed by dividing the video into keyframes. Then object detection algorithms are used to extract the most important items appearing in the key frames [1]. Finally, the composition of all selected objects into one image is performed to create the final static image summary (see Fig. 1).

![Fig. 1 Video summary example of relevant faces](image)

When designing the summary, the research questions that we address are:

1. How good can a single image representation of video content become?
   
   A single image output is a requirement of the system. However, it is not the only requirement we want to fulfill. The image must browse the video and sort its content. A second question is therefore:

2. Which is the best method to compose the resulting object map?
   
   We do not want to create an object map with randomly positioned items, but rather generate a self-explanatory map which may be used by users for browsing the video. As can be seen, users’ opinion is very important for the thesis. We focus our last two questions on them:
3. Which content may be selected for the user to understand a video?

Content selection is very important for creating a good representation. We will analyze user attention models approaches [2] to detect where we would find users’ regions of interest. The approach should be validated by verifying whether its results fulfill the original user requirements. Evaluating video summarization is a difficult but important problem:

4. How can we evaluate the video summarization results taking into consideration the point of view of the user?

An evaluation process is performed to validate our approach by means of a user study. We present the motivation of the thesis in next section as well as analyzing different application fields in Section 1.3.

1.2 Motivation

Today, video summaries are based on textual descriptions of video content, such as duration, type, authorship and relevance of the video. These data does not always give enough information to the users and they have to browse the video content in order to determine if it is relevant or not.

Another type of video summarization is video skimming. A video skim is a temporally compacted form of video stream that should preserve the most important information. As synonyms to video skim, researchers have used the terms preview and trailer in the literature.

Finally, other summarization systems are based on keyframe representations of the video content. With these methods, multiple keyframes should be used in order to generate a complete representation of the whole video. However, Dufaux presents a method to automatically extract a single image representation as a summary analyzing semantic content and movement in the video scenes as a variant of keyframe-based summarization [1].

With object mapping we merge the information in keyframes, content-based video analysis and the simplicity of static story-board summarization. Object maps can give complete and compacted information of the video content to the user as well as methods for rapidly navigating through the original video, thus giving him the opportunity to select the important parts. Some interesting applications are explained in the next section.
1.3 Applications

Object mapping has numerous applications for video navigation, search and database management, and it helps to include hyperlinks into existing content. A quick visualization of the video content helps users to rapidly detect if it is relevant. Regarding large video databases, for example, it may reduce significantly the time required for searching a specific content or a specific video.

Video navigation is another application for our approach. With a static image, users can use the object map as a visual index that will allow a fast access to the shot from which each object was extracted without using sliding bars or other techniques, only with a simple click.

Furthermore, the visual representation complements textual metadata of the video, not only general video metadata, but metadata related to each represented region in the map by defining clickable areas within it. Who is the actress? What model is that car? Where can I buy it? Does it appear in other moments of the video? These are some questions that the provider of the summary would want to add as textual metadata, links to the stores selling the object and more.

Finally, the proposed approach can also be useful for automatic indexing applications because the selected regions may be the only ones processed by pattern recognition algorithms. This way, the object mapping technique would be understood as a pre-processing step that selects a small subset of regions to be processed by other image processing techniques. For example, if automatic indexing system contains a face recognizer for actors/actresses in the video, evaluating it in every single frame of the video is not needed, but only on the selected regions included in the object maps. By doing so, the required computational effort could be reduced dramatically.

1.4 Outline of the thesis

The rest of the thesis is structured as follows: In Chapter 2 we describe different techniques used for video summarization, some of them are shorter video representations (video skims). We then describe other methods based on static image representations. They will be deeply analyzed mentioning face and object detection algorithms to extract semantic content from the video.

In Chapter 3 we analyze the system requirements as well as the priorities to get them in terms of user’s acceptance of the proposed summary. Then, in Chapter 4
we propose our approach. Domain knowledge using movie trailers is applied to analyze the relevance of the content included in the video summary. The composition of the final mapping is performed using this knowledge, but also the architecture can be customized using trained object detection methods. Our solution is evaluated in Chapter 5 by means of a user study, and in Chapter 6 and 7 we discuss our conclusions and future work.
2

Related work

In this chapter, we describe the video summarization techniques employed today to achieve new levels of understanding. We begin in Section 2.1 with an explanation of existing types of video summarization techniques. In subsequent sections we explain the workflow of the process and the involved technologies. In Section 2.2 we discuss the temporal segmentation methods that researchers use. In Section 2.3 we explain different content selection techniques used by the community in order to detect important video segments to be included in the video summary. Finally, in Section 2.4 we present object extraction methodologies for the correct understanding of the architecture of our final approach for video summarization.

In Section 2.1 we define video summarization terminology used in related literature. We also describe briefly existing summarization techniques in order to understand how this chapter is divided in subsequent sections.

2.1 Definitions

Video summarization engages in providing concise and informative video summaries in order to help people browsing and managing video objects more efficiently. It has received more and more attention in recent years because new
utilities (social networks, portable devices, etc.) allow users to access video content easily but they need to manage this content properly. Basically, there are two different kinds of video summaries: static image summary and moving-image skimming.

2.1.1 Moving-image skimming

The moving-image skimming, also known as video skim, consists of a collection of video clips, as well as the corresponding audio segments extracted from the original sequence and is thus itself a shorter version of the original video [3] [2] [5]. They can be classified into two types: Overview and Highlight.

In the classic case of movie trailers, the user is usually unaware of the content and is interested in a much reduced summary of the video content to decide before watching the full versions. We call this kind of video skimming overview. For a specific domain like news or sports, users want to see the most important events in the video (goals, news headlines) according to their interests. This type is called highlight. Unlike overviews, which are presented as single condensed videos, highlight-based summaries are usually presented as an organized list of interesting events along with some associated metadata.

2.1.2 Static summaries

The static summary, also known as storyboard, is a small collection of salient images or a single one extracted or generated from the underlying video source [15] [20] [21] [25]. According to the method used to extract representative images, we can classify static video summaries into sampling-based, shot-based, motion-based, mosaic-based and object mapping methods.

Sampling-based methods select video keyframes by random or by uniform sampling of the input video. For shot-based methods, the source video is temporally segmented into shots using shot boundary detection algorithms. Motion-based methods refer to the temporal dynamics of the video by motion analysis using image pixel differences or optical flow. When the camera motion can be detected, a mosaic image can be constructed to represent the whole content of a dynamic shot. Finally, object mapping aims to extract relevant objects from the source video to create a composite image.
In the next sections we review different technologies used to construct the commented summaries as well as techniques to temporally segment the source video and to extract relevant object content (faces, cars, etc).

### 2.2 Shot segmentation

Temporal redundancy is a very important issue that needs to be solved when facing video processing. Deleting redundant information is achieved by segmenting the video into shots. A shot is a continuous recording of video content without breaks in a scene. Then, keyframes may be extracted from each shot with different techniques based on pixel-to-pixel comparison, histogram-based comparisons, motion flow vectors, etc. This process is called *Shot Boundary Detection* (see Fig. 2).

![Fig. 2 Shot boundary detection example](image)

Pixel-to-Pixel methods are the core methods and probably the most straightforward ones [11]. Indeed, the first idea that comes to mind when we want to compare two images in terms of similarity is to compare their pixels.

Histogram-based methods get better reflection of global properties of a picture, which is their main advantage [12]. These techniques are significantly more robust against camera and object motion. However, there are drawbacks: a shot boundary occurring in two frames with similar histograms will be missed; also, significant luminance difference between frames will declare false positives in shot boundary detection.

Histograms may be compared in different ways [13]. A first approach would be to calculate the histogram of each color channel that form the image and, then, calculate the difference between the bins in each histogram of the two successive
frames. Another technique is to calculate the difference of all channels between the histograms in the two images and take the maximum to the sums in order to detect significant changes in one channel. Finally, a variation of the last technique is to weight the importance of each color channel.

A method that uses Hausdorff approximation to determine the outliers is used in [13]. The Hausdorff method performs an edge detection process of the image and compares the location of the edge points produced by the edge detector. The method checks for each point whether a corresponding edge exists in the successive image. If the sum of non-correlated edges is greater than some threshold, a shot boundary is declared (see Figure 3).

![Fig. 3 Shot detection example using Hausdorff distance method](image)

[13] also presents a combination of all the commented methods by building an ensemble method, Neural Network (NN). The inputs are the outputs of the different methods with a supervised learning process to easily adapt results for different type of videos. Weaknesses of each method are compensated by the others and the NN is adapting to any given threshold by propagating the errors to its weights.

More recent techniques include a higher-level segmentation of videos into scenes. Rasheed and Shah [14] present a method based on a graph partitioning problem that clusters shots into scenes constructing a graph called shot similarity graph (SSG). Each node represents a shot and the edges between them are weighted based on their similarity based according to color and motion information. Then, the SSG is split into sub-graphs by applying normalized cuts representing individual scenes. They also propose a method to describe the content of each scene by selecting a representative keyframe.

To sum up, there exist several shot segmentation techniques:

- Simple approaches compare pixel intensity and image histogram to decide whether two frames belong to the same shot.
- Later approaches include edge evaluation and comparison between frames using Hausdorff distance.
- Learning processes using NN are also used to adapt the shot detection to the source video regardless of thresholds.
- Recent techniques use clustering methods to group similar frames based on pixel color, motion flow information, etc.
In the next subsections we describe in detail the approaches used in the thesis. Our prototype is explained in Chapter 4.

### 2.2.1 Software Initiative Studies at UCSD

In this approach, each frame is divided into NxN regions. Then, the pixel change is estimated for each region and pairs of frames. If the pixel change is greater than some threshold and its cumulative sum is greater than the region threshold for the frame, then it triggers the shot boundary detection. This technique also provides a simple frame averaging to avoid luminance changes that could be detected as a shot boundary. This pixel-to-pixel method combines low computational requirements with satisfactory results, but also tends to generate some false detection, which generate an over-segmentation of the video (see Fig. 4).

![Fig. 4 Shot boundary detection using UCSD pixel regions difference](image)

### 2.2.2 Course Project of Binshtok and Greenshpan at BGU

A second software kit has also been tested resulting from a course project by Max Binshtok and Ohad Greenshpan [13], two students at the Ben-Gurion University of the Negev (BGU) in Israel. The proposed software includes four different algorithms for the shot boundary detection: a pixel-to-pixel method, a histogram-based method, a third one based on the Haussdorf distance, and a learning process based on NN.

While pixel-to-pixel methods might not be state of the art, they work quite well for the movie trailers we aim at processing in our Thesis considering typical fast-paced trailer editing. The classic solutions that segment shots based on motion estimation features do not provide different views of the same object or faces, a feature which is desirable to build the object maps by selecting the best view of...
every object. The pixel-to-pixel method naturally generates over segmentations of the videos due to changes in luminance or points of view as shown in Fig. 5.

![Fig. 5 False shot detection useful for the project. Frontal and side views](image)

There are many types of pixel comparisons used in the approach:

- **Global Pixel-to-Pixel**: This method sums the pixels’ intensity values over the whole image, and compares it to the sum of the pixels’ intensity values in the second image as shown in equation (1).

\[
\frac{\sum_{i=1}^{X} \sum_{j=1}^{Y} I(t, i, j) - \sum_{i=1}^{X} \sum_{j=1}^{Y} I(t + 1, i, j)}{256 \cdot XY} > \tau
\]

(1)

\(I(t, i, j)\) represents the intensity value of pixel \((i, j)\) at time \(t\). If the difference is bigger than some threshold \((\tau)\), a shot detection is declared. It is obvious that the local differences between pixels’ intensity values are ignored.

- **Cumulative Pixel-to-Pixel**: This method sums the differences between each pixel’s intensity value in one image and its intensity value in the successive image. We take into consideration local details in the images as shown in (2).

\[
\frac{\sum_{i=1}^{X} \sum_{j=1}^{Y} | I(t, i, j) - I(t - 1, i, j) |}{256 \cdot XY} > \tau
\]

(2)

The histogram-based methods compare the pixel histograms of neighboring frames to determine the shot boundaries. They introduce robustness against camera and object motion, but they fail at segmenting two shots whose colors are similar. Important methods are:
• **Simple histogram**: This method calculates one histogram per color channel that form the image and compute the difference between the bins in each histogram of the two images using (3).

\[
\frac{\sum_{C \in \{\text{channels}\}} \sum_{b=0}^{\text{Bins}} |H(t,c,b) - H(t-1,c,b)|}{\text{pixels} \cdot |\text{channels}| \cdot 2} > \tau
\]

(3)

\(H(t,c,b)\) represents the histogram value of the bin \(b\) in the color channel \(C\) at time \(t\).

• **Max histogram**: This method calculates the differences over all channels between histograms in the two images and takes the maximum of the sums. It can be influenced by an intense change in one channel as shown in formula (4).

\[
\max_{C \in \{\text{channels}\}} \frac{\sum_{b=0}^{\text{Bins}} |H(t,c,b) - H(t-1,c,b)|}{\text{pixels} \cdot 2} > \tau
\]

(4)

• **Weighted histogram**: It also takes into account the histograms’ difference in all channels and gives each one a weight, determined by luminance proportions of the channel, thus giving more weight to the prevalent color channel in the image as shown in (5).

\[
\sum_{C \in \{\text{channels}\}} \sum_{b=0}^{\text{Bins}} \frac{w_c}{w_{\text{mean}}} |H(t,c,v) - H(t-1,c,v)| \frac{\text{pixels} \cdot |\text{channels}| \cdot 2}{\text{pixels} \cdot |\text{channels}| \cdot 2} > \tau
\]

(5)

The Hausdorff method performs an edge detection process with the Sobel operator on the images and compares the locations of these points between frames. It is a good approximation to get the same face or object twice if there exists any smoothing.

Finally, Binshtok and Greenshpan’s paper states that the option that combines the three methods using a neural network provides the best results for the keyframe extraction. This is because preset input thresholds (\(\tau\)) do not play role in the shot boundary detection. Instead, the NN adapts to any given value by propagating the errors to its weights.
2. Related work

Fig. 6 Shot boundary detection using NN

In the next section we analyze different content selection techniques that were proposed by researchers. The better the temporal segmentation and shot boundary detection are performed, the less redundant information should be processed and the greater the performance of the content selection methods will be.

2.3 Content selection

In this section we analyze different approaches for the content selection included in different video summaries. We begin with video skim generation in Section 2.3.1 and continue with reviewing techniques used for static image summaries in Section 2.3.2.

Early attempts did not use content analysis but image processing techniques that, in most cases, make the result non self-explanatory and do not possess a well-defined structure representation. Over the years the trend changed to including well balanced content extraction and video structure. The problem of most traditional summary generation approaches is that they are based on low level features. Hence, they may not be able to guarantee that generated results include relevant content. Many attempts try to deal with this problem but they are mostly the highlight generation approaches. That means that the video category has to be known in order to obtain relevant content. Such methods may not be used on generic videos.
2.3.1 Dynamic video skimming

A dynamic video skim consists of a collection of audio-video sub-clips. It preserves the dynamic properties of the original video. In [3] frames with high-contrast are detected as the ones containing important content. Furthermore, calculating frame-to-frame differences lets them extract high-action parts from the video. In addition, the average color composition of the whole video is considered to include similar frames in the video skim. Finally, spectrum of simple alphabetic characters for dialog recognition is performed.

Another simple approach based on time compression technology is [4]. It allows faster playback of the video when playing static video scenes and slower speed for short and dynamic video scenes. It uses an audio time scale modification technology to preserve the comprehensibility of speech. However, the maximum time compression depends on the speed of speech. Also, this approach distorts original video temporal property and it does not include content analysis.

The Informedia project [5] [6] [7] creates the summary by extracting significant audio and video information. Text keywords from captioning and manual transcript are first extracted using a term-frequency – inverse document frequency [51] technique. This text is used to create a skimming version of the audio including some neighboring segments for better comprehension. Then, the image skimming is created by selecting frames with a descending priority: frames with faces or texts, static frames following camera motion, a combination of frames with camera motion, faces and texts, and frames at the beginning of a scene are selected. This synopsis is not aligned with the audio in time and it cannot be used with videos with more complex audio content (music, audio effects). Even so, both audio content and content analysis achieve impressive results.

A method to generate video skims based on user attention model is presented in [2] (see Fig. 7). Attention is described as a neurobiological conception that implies the concentration of mental powers on an object or audio track. Computing attention allows authors to avoid the problem of semantic understanding of the video content. The attention modeling includes visual, audio, and text modalities that together generate the user attention curve. Hence, an attention value is assigned to each frame to determine which of them are more attractive for the viewer and thus constitute the summary.
2. Related work

![Framework of user attention model](image)

This method [2] constructs a video summary without fully semantic content understanding. However, humans do not only understand videos not by perceiving just these low level features. Also, the fusion scheme of the attention model has to be improved because it is not proved whether it is the most effective and the information on the video structure is neglected.

[8] proposed an approach for summarization that emphasizes both the content balance and perceptual quality of the summary. A clustering method is used to cut the video, and a motion attention model is used to assess the perceptual quality of shots and clusters. Both together create a temporal graph that describes the evolution and importance of the clusters. This temporal graph is utilized to group scenes from clusters while the attention values aim to select the appropriate scenes for summarization.

Another method for the creation of video skims based on similarity between shots is presented in [9]. A combination of Hausdorff distance and Boolean model is used to compare shot similarity. Then, a shot clustering is performed with the affinity propagation clustering method [10] and, finally, content ranking is added to select shots included in the video summary. With shot similarity, clusters can be created that reduce the redundancy of the summary and thus achieve good compression ratios. With partial semantic understanding, high user satisfaction and informativity are achieved in experimental results.

To sum up, the techniques used to generate video skimming summaries have evolved as follows:

- First approaches aim at providing summaries without analyzing the content. They select important scenes based on low-level image properties.
- Later approaches provide scenes based on important content. Content selection techniques involve both user attention studies and relevant object detection.
- Other approaches use clustering methods to measure shot similarity and add the most diverse clusters to reduce the redundancy in the resulting summary.
In the next section static video summary generation literature is analyzed. Most described approaches use similar methods to select the content to be included in the summary. The main difference between this section and the next one is the content presentation and the importance of compacting information into one single image or a group of static representations.

2.3.2 Static video summarization

A static video summary can be expressed in a collection of images or a single one that represents the video content. Early work [15] selects video keyframes by random or uniform sampling of the video image sequence. These methods are simple and they are unable to guarantee that important content is covered in the result.

Shots are important building blocks of videos. They represent a continuously captured sequence, and shot transition detection has been suggested in various works [16] [17] [18] [19] since its visual content can be represented by some frame. These methods extract mostly the first frame of the shot, but in [17] subsequent frame histograms are computed. Once the difference exceeds a certain threshold, a new keyframe is extracted to be included in the summary.

In [20] a scene transition graph is constructed for a video by time constrained clustering on the video shots. In it each video shot cluster is represented by one node in the graph and the transitions between nodes reflect the structure of the video.

In [21] an unsupervised clustering scheme is proposed to extract the keyframes. First, all frames are clustered based on the color histogram similarity comparison into a certain number of clusters with a predefined threshold. Next, all clusters that are big enough to be considered important a representative frame is selected as the closest to the cluster centroid from each of them. The system is robust against background noise and motion but its performance highly depends on the threshold selection.

Later works concentrate on organizing shot images by analyzing the video structure since videos comprise many video shots. In [22] the video content is represented in a tree structure. From top to bottom, a video consists of several scenes; each scene is composed of several related shot groups. Each shot group is composed by several visually similar and temporally adjacent shots. This tree structure represents an abstraction of the video content and it is presented to the user as a summary.

[23] creates a mosaic image to represent the whole content of a dynamic video shot if the camera motion can be detected (pan, tilt, zoom, translate). Although this approach is informative, it only provides an extended panoramic spatial view of
the entire static background, but contains no information about the moving foreground. In the situation that the scene is changing frequently and the camera motion is complex, this algorithm tends to achieve only poor results.

![Mosaic representation](image)

**Fig. 8 Mosaic representation. Top: Hand-chosen keyframes. Bottom: Mosaic representation without foreground occlusions [23]**

Recent works present effective methods for summarizing relevant content. A comic book style video summary is generated in [24] such that the size of selected images is adjusted according to their importance. The output video structure reflects how the editor chooses and arranges video shots; they represent very valuable information for video summarization.

[25] provides a static video summary consisting of three major procedures: keyframe extraction; estimating regions of interest (ROI) from extracted keyframes, and assembling the ROI into one image by arranging them according to the temporal order and their size. The proposed method generates expressive video summaries and conserves both plot and temporal information.

*Video Summagator* (VS) [26] is a volume-based interface for video abstraction and navigation. VS models a video as a space-time cube and visualizes it using real-time volume rendering techniques. The project also empowers the user to manipulate the video cube interactively to not only understand the content but also to navigate the content of interest.

The approach of Chen in [27] extracts and visualizes movie storylines in a static image in order to get a quick overview. *Visual Storylines* preserve the elegance of original video by a sequence of video analysis, image synthesis, relationship quantification and geometric layout optimization techniques. They cluster video shots according to both visual and audio data to analyze and quantify story relationships. A multi-level storyline visualization method organizes both location and interesting objects and characters (see Fig. 9). This kind of representation can be used to assist viewers to grasp video content efficiently, especially when a text synopsis is provided. Highly condensed video summarization techniques in which keyframes are packed and visualized using irregular shapes [28] have a common problem: due to their highly compact form and losses of information it is nearly impossible for viewers to extract stories. *Visual Storylines* solves this problem of [29] by visualizing the information of locations and relations between interested objects.
To sum up, static summary generation merges methods similar to video skimming with different compositing techniques to correctly plot all selected information:

- Static summarization techniques are based on a single keyframe representation.
- Early approaches represent important content in a mosaic and comic-based representation that includes more information than a single image representation (see Fig. 8).
- Space-time cube representations are used in later approaches to provide navigation utilities for users.
- Finally, storylines are composed into an image that aims at providing a fast understanding of the video content.

For our approach, we focus on the most relevant content to build our summary, faces. The next section describes in detail several content selection techniques that can be used to extract relevant content. Then, features used to extract relevant content from source videos are described in detail.
2.4 Object detection

Object detection, and especially face detection, has been a core problem in computer vision for more than a decade. Not only has there been substantial progress in research, but many techniques have also made their way into commercial products.

The Viola-Jones [30] machine learning approach for visual object detection is capable of processing images extremely rapidly and achieving high detection rates. It is distinguished by three key contributions to the object detection field: the integral image, AdaBoost machine learning and cascade generation that combines increasingly complex classifiers. It is well-established, scale-invariant and works fast. However, it is not rotation-invariant and requires long training time.

Another approach that tries to reduce computing time and to solve rotation variation are the Speeded-Up Robust Features (SURF) [31]. SURF finds interest points in the image using Hessian matrices of the interest points, determining their orientation, and using Haar wavelets in a square region around the interest points to find intensity gradients. The matching process is done by comparing references to the query image features. No training process is needed and objects can be detected in real-time.

There exist other visual features that should improve the object classification performance. Eigenfaces and Fisherfaces treat the visual features as a vector in a high-dimensional image space [32]. Working with high dimensions is costly and unnecessary in real-time applications. The Eigenfaces approach maximizes the total scatter, but it is a problem in an unsupervised scenario because the detection algorithm may generate faces with high variance due to the lack of supervision in the detection. Although the Fisherfaces method can preserve discriminative information with Linear Discriminant Analysis, this assumption basically applies for constrained scenarios. Some frameworks cannot guarantee a training set of images from the same person/object, so the estimated covariance for the subspace may be inferior. For this reason [33] propose Local Binary Patterns Histogram (LBPH), a method that extracts local features and focus on 2D texture analysis to create low-dimensional features, trying to preserve the useful information. In this way we can create a method for object detection that does not require training process with large training dataset.

When the objects are observed from multiple viewpoints and unconstrained scenarios, the detection task becomes harder [34]. The common practice is to divide into subcategories. For instance, faces can be categorized as frontal, right/left profile, rotation, etc.

Different classifiers can be trained for different subcategories. In [35] [36] a pose estimator is first built to classify each example into one subcategory. Each
subcategory trains its own classifier for detection with manually labeled data. It is a very laborious and difficult task for other kind of detections such as cars. The cluster boosted tree classifier [37] applies a conventional $k$-means clustering algorithm to split samples when learning rates are low. The authors show that by using previously selected features for clustering, the learning algorithm converges faster and achieves better results.

The misclassification caused by the pose estimator is one weakness of the [36] method. It also happens in training caused by mislabeling. It is possible that the boundary between two viewpoints can be very subtle and differs for different individuals. Furthermore, traditional training processes lack the flexibility to re-categorize examples during training. In [38] multiple category learning is proposed to solve this problem through adaptive labeling. The winner-take-all multiple category boosting algorithm learns all subcategory classifiers simultaneously with the assumption that the final classification of an object will only be determined by the highest score of all subcategory classifiers. Subcategory labels are dynamically assigned in this process, thus reducing the risk of having outliers.

A method that has become quite popular is the discriminatively trained, multiscale, deformable part model for object detection [40]. As described in the related literature, detection with deformable part models can be done by considering all possible variations of a distinguished “root” part and, for each of those, finding the best configuration of the remaining parts (see Fig. 10). In [41] a general method for building cascade classifiers from these models is described. Star-structured models are primarily focused as well as partial hypothesis pruning to speed up object detection without reducing detection accuracy. The authors introduce probably approximately admissible thresholds that provide theoretical guarantees on the cascade performance and can be computed from a small sample of positive examples.

Fig. 10 Deformable part model detection [41]
To sum up, the object detection methods have evolved into different parts or views training processes to improve the performance:

- Early approaches training processes did not take into account different views of the object, thus, they do not perform well in unsupervised environments.
- SURF features solve this problem by selecting points of interest in the image and calculating their rotation and position within the training image.
- There exist also other solutions like LBPH, Eigenfaces, or Fisherfaces that combine both learning and matching using different features instead of Haar and SURF features.
- Dividing the training process into multiple viewpoint is a recent approach to improve rapid object detection performance.
- Finally, to correctly detect an object, a part-based detection process is performed in most recent approaches.

In the next subsections we describe in detail the algorithms we have used in the thesis to extract relevant content as well as different features.

### 2.4.1 Haar-based cascade classifiers

The work proposed by Viola and Jones [30] has shown satisfactory performance for simple viewpoint object detection tasks and was improved by R. Lienhart [42]. It combines four key concepts: Haar features, integral images, AdaBoost machine learning and cascade classifier generation.

Used features are not true Haar wavelets but simple rectangular features. They contain better suited rectangle combinations used for visual object detection. The presence of a Haar feature is determined by subtracting pixel values of the dark region to pixel values of the bright one. If the difference exceeds some threshold set during the training process, the feature is assumed to be present.

![Fig. 11 Haar-like features used in OpenCV](image_url)
The first two features selected by the described approach are shown in Fig. 12. The two features are shown in the top row and then overlaid in a typical training face in the bottom row. The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. It employs the observation that the eye region is often darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose.

Fig. 12 First two features selected in Viola-Jones algorithm

Feature computation requires summing pixel values covered by the rectangles. This addition can be very efficiently performed with the integral image, also known as Summed Area Table.

Fig. 13 Summed Area Table example

The integral image at location $x, y$ contains the sum of the pixels above and to the left of $x$ and $y$:

$$ ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') $$

(6)

where $ii(x, y)$ is the integral image and $i(x, y)$ is the original image. In the example shown in Fig. 13, the sum of the pixels within the green rectangle can be computed with four array references: the value of the integral image at location A is the sum of the pixels in the red rectangle, 5. The value at location B is 5 + 2, at location C is 5 + 3, and at location D is 5 + 2 + 3 + 6. Then, the sum of the original image pixels within the green rectangle can be computed as $16 + 5 - (7 + 8) = 6$.

The AdaBoost machine-learning method combines many weak classifiers (stages) that give the right answer more often than a random decision to create
strong classifier. That is because the training error of the strong classifier converges to a minimum over time. In their approach, a variant of AdaBoost is used both to select small set of features and to train the classifier.

The weak learning algorithm is designed to select a single rectangle feature from the 180,000 potential features, which best separates the positive and negative examples. For each feature, the weak learner determines the optimal threshold classification function, such as the minimum numbers of examples that are misclassified. Hence, a weak classifier $h_j$ consists of a feature $f_j$, a threshold $\theta_j$ and a parity $p_j$ indicating the direction of the inequality sign:

$$h_j = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

(7)

where $x$ is a pixel sub-window of an image. In practice, no single feature can perform the classification task with low error. Features which are selected in early rounds of the boosting process usually have lower error rates (0.1 to 0.3) than features selected in later rounds (0.4 and 0.5), because the task becomes more difficult.

Finally, a cascade of classifiers is constructed to achieve increased detection performance. AdaBoost gives weights to each stage and sets the order of filters in the cascade. The higher weighted filter comes first to eliminate non-face regions as soon as possible.

![Fig. 14 Cascade classifier for face detection](image)

A cascade is a degenerated decision tree (see Fig. 14). A positive result from the first classifier triggers the evaluation of a second classifier which has also been adjusted to achieve very high detection rates. A positive result from the second classifier triggers a third one, etc. A negative outcome at any point leads to the immediate rejection of the sub-window.

Stages in the cascade are constructed by training classifiers using AdaBoost and then adjusting the threshold to minimize false negatives. The default AdaBoost threshold is designed to yield a low error rate on the training data and, in general,
lower thresholds yield higher detection rates. Then, an excellent first stage can be constructed by reducing the threshold to minimize false negatives.

The structure of the cascade reflects that, in any single image, a majority of sub-windows are negative. Hence, the cascade attempts to reject as many negatives as possible at an early stage. A positive instance that triggers the evaluation of every classifier in the cascade is a rare event, subsequent classifiers are trained using those examples which pass through all the previous stages. As a result, every classifier is more difficult than the previous one and, at a given detection rate, deeper classifiers have higher false positive rates.

The cascade training process involves two types of tradeoffs. Classifiers with more features will achieve higher detection rates and lower false positive rates but they require more computing time. An optimization framework can be defined with: the number of stages, the number of features in each stage, and the threshold of each stage. It is a sophisticated problem, because in practice each stage reduces the false positive rate and decreases the detection rate. Each stage is trained by adding features until the target detection and false positive rates are met and stages are added until the overall target for false positive and detection rate is reached.

2.4.2 Speeded-Up Robust Features

SURF [31] is a scale and rotation-invariant interest point detector and descriptor. It outperforms previously proposed schemes with respect to repeatability, distinctiveness, and robustness. It also can be computed and compared much faster. It is partly inspired by the SIFT descriptor [43] but the standard version is several times faster than SIFT and claimed to be more robust against different image transformations.

The described detector is based on the Hesse matrix because of its good performance in computation time and accuracy. Given a point \( X = (x, y) \) in an image \( I \), the Hessian matrix \( H(X, \sigma) \) in \( X \) at scale \( \sigma \) is defined as follows:

\[
H(X, \sigma) = \begin{bmatrix}
L_{xx}(X, \sigma) & L_{xy}(X, \sigma) \\
L_{xy}(X, \sigma) & L_{yy}(X, \sigma)
\end{bmatrix}
\]

where \( L_{xx}(X, \sigma) \) is the convolution of the Gaussian second order derivative with the image \( I \) in point \( X \), and similarly for \( L_{xy}(X, \sigma) \) and \( L_{yy}(X, \sigma) \). Gaussians are optimal for scale-space analysis and it is discretised and cropped (see Fig. 15, left half). The 9x9 filters in Fig. 15 are approximations for Gaussian second order derivatives.
2. Related work

Scale spaces are usually implemented as image pyramids. The images are repeatedly smoothed with a Gaussian and subsequently sub-sampled in order to achieve higher levels of the pyramid. Using box filters and integral images, they do not have to iteratively apply the same filter to the output, but only apply such filters of any size at exactly the same speed on the original image. Hence, the scale space is analyzed by up-scaling the filter size rather than iteratively reducing the image size.

In order to extract interest points in the image and over scales, a non-maximum suppression in a 3x3x3 neighborhood is applied. The maxima of the determinant of the Hessian matrix are then interpolated in scale and image space.

The proposed SURF descriptor is based on similar properties of SIFT. The first step consists of an orientation assignment calculating Haar-wavelet responses in x and y direction (see Fig. 16). Then, a square region is constructed pointing at the selected orientation, and finally SURF descriptors are extracted.

For the extraction of the descriptor, the first step consists of constructing a square region centered around the interest point, and oriented along the selected orientation. The region is split up regularly into smaller 4x4 square sub-regions. Each sub-region has a four-dimensional descriptor vector $v$:

$$v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$$

$d_x$ is the Haar wavelet response in horizontal direction and $d_y$ the Haar wavelet response in vertical direction. Both directions are defined in relation to the selected interest point orientation to increase the robustness towards geometric deformations and localization errors.

The next figure helps to understand the properties of the descriptor for three distinctively different image intensity patterns within a sub-region. In case of a homogeneous region, all values are relatively low. In presence of frequencies in x
direction, the value of $\sum |d_x|$ is high, but all others remain low. Finally, if the intensity is gradually increasing in $x$ direction both values, $\sum d_x$ and $\sum |d_x|$, are high.

![Fig. 17 SURF descriptor performance in different image intensity patterns [31]](image)

### 2.4.3 Local Binary Pattern Histograms (LBPH)

Unlike Eigenfaces and Fisherfaces, LBPH extracts local features of the object and has its roots in 2D texture analysis [33]. The basic idea of LBP is to summarize the local structure in a block by comparing each pixel with its neighborhood. Each pixel is encoded as a sequence of bits, each of them associated to the relation between the pixel and one of its neighbors. If the intensity of the center pixel is greater-equal to that neighbor’s, then encodes the relation as 0; encodes as 1 otherwise (see Fig. 18).

![Fig. 18 LBP code creation example](image)

At the end, a binary number (LBP code) is created for each pixel. If 8-connectivity is considered, we end up with 256 combinations. This histogram-based approach defines a feature which is invariant to monotonic grayscale transformations as shown in Fig. 19.

![Fig. 19 LBP invariant to monotonic grayscale transformations](image)

---

1 Images extracted from OpenCV documentation. Last visit on May, 28th 2013.
The spatial information must also be incorporated in the face recognition model. The proposal is to divide the LBP image into 8x8 local regions using a grid and extracting a histogram from each. Then, the spatially enhanced feature vector is obtained by concatenating the histograms, not merging them.

These features have low-dimensionality but they are not robust to variations in illumination, scale, translation or rotation. For these reasons, it is extremely important to apply image processing techniques to standardize the input block (size, histogram equalization, etc).

### 2.4.4 Deformable parts-based cascade classifiers

[41] describes an object detection system based on mixtures of multiscale deformable part models. Deformable part models have become quite popular because they provide a solution to the problem of detecting and localizing generic objects from categories that can vary greatly in appearance such as people or cars.

While deformable models can capture significant variations in appearance, a single deformable model is often not expressive enough to represent a rich object category. Even so, simple models can perform better in practice because rich models often suffer from difficulties in training. For object detection, rigid templates can be easily trained using discriminative methods but richer models are more difficult to train, in particular, because they often make use of latent information.

The part-based model used in this approach is star-structured defined by a root filter plus a set of parts filters and associated deformation models. The detection score of the model can be calculated as follows:

\[
\text{score}(\text{model}, x) = \text{score}(\text{root}, x) + \sum_{p \in \text{parts}} \max[\text{score}(p, y) - \text{cost}(p, x, y)]
\]

The score at a particular position and scale within an image, \(\text{score}(\text{model}, x)\), is the score of the root filter at the given location plus the sum over parts of the maximum, over placements of that part, of the part filter score on its location minus a deformation cost measuring the deviation of the part from its ideal location relative to the root. Fig. 20 shows a star model of a person category where (a) is the root filter, (b) are several higher resolution part filters, and (c) a spatial model for the location of each part relative to the root which reflects the “cost” of placing the center of a part at different locations relative to the root. The filters specify weights for histogram of oriented gradients (HOG) features.
The training process uses a latent SVM (LSVM) to train models using partially labeled data, learn $\beta$, and data-mining for hard negative examples. Each example $x$ is scored by the following function:

$$f_\beta = \max_{z \in Z(x)} \beta \cdot \Phi(x, z)$$

(10)

Here, $\beta$ is a vector of model parameters, in the case of a star model it is the concatenation of the root filter, the part filters, and deformation cost weights. The $z$ are latent values, and $\Phi(x, z)$ is a feature vector, a concatenation of sub-windows from a feature pyramid and part deformation features. $Z(x)$ is a set of possible latent values for $x$. Parameter $\beta$ is learned by minimizing the next function:

$$L_D(\beta) = \frac{1}{2}||\beta||^2 + C \sum_{i=1}^{n} \max (0.1 - y_i f_\beta(x_i))$$

(11)

with $D = \langle (x_1, y_1), \ldots, (x_n, y_n) \rangle$, is a set of labeled examples.

The loss function is convex in $\beta$ for negative examples and $L_D(\beta)$ is convex when latent variables are specified for positive examples. If there is a single possible latent value for each positive example $f_\beta$ is linear.

To detect objects in an image the authors of [41] compute an overall score for each root location according to the best possible placement of the parts. Parts that yield a high-scoring root location define a full object hypothesis. Dynamic programming and generalized distance transforms are used to compute the best locations for the parts as a function of the root location.
Fig. 21 illustrates the matching process at one scale. Responses from the root and part filters are computed at different resolutions in the feature pyramid. The combined scores clearly show two good hypotheses for the object at this scale.

Finally, the detection results for one object show a considerably amount of overlapping detections. The chosen solution is to sort detections by score. They add the detection one by one and skip those which have detection overlaps of 50% or more.
3

Requirements

As we have seen in the previous chapter, video summarization systems can serve different purposes; they can be developed for specific types of content or different types of users. In Section 3.1 we narrow down the scope of this thesis and we analyze the requirements from the users’ perspective in Sections 3.2 and 3.3.

3.1 Scope of the thesis

In this thesis we address the problem of designing an automatic system with existing algorithms that can create efficient image representations of video items to help users detect important objects in the video as well as to provide a quick navigation through it. Our aim is to design a system that can automatically create a high quality video summary from a content source video.

For testing, we use commercial movie trailers as source videos. They are one of the main advertising tools of the movie industry. They are not made to give a fair impression of a film, but rather to convince people to watch the movie; they are constructed with a proper onset time of main characters, important locations and relevant objects inside the movie. We make use of this marketing strategy to select the important content the user wants to see in the summary.
3. Requirements

How do people actually choose what to watch? People tend to read the summaries of the movies. Another very interesting approach is to select those movies in which your favorite actor/actress appears. In TV watching behavior, for example, people consult online program guides during viewing to look for further information.

A rapid grasp of the video content is a valuable source of information for most users. Allowing users to browse different moments of the source video easily through a relevant content representation allows them to temporally navigate avoiding extensive search efforts on the full video and without sliding scroll bars.

Since the relevant content will be available in the final summary, additional textual metadata should be added. Each object within the summary may be described accessing the web. For instance, if Brad Pitt appears in the movie, a user may want to see his filmography in order to know in which other movie he has starred. This can be applied to objects too. Where can I buy this car? What are its technical specifications? The provider of the summary would want to add this information hyperlinking all these regions in the image. This leads to the central question: “What content is relevant for users?”

In the next section we try to answer this question by presenting a set of requirements that the video summary should fulfill.

3.2 Requirements analysis

Our user requirements for fast and convenient content selection are mostly derived from related literature on video summarization (see Chapter 2 and [2] [27] [25]). The outcome of this analysis is a list of ten requirements grouped in four categories: priority, uniqueness, structural and navigability.

*Priority* requirements specify what type of content should be preferably included in the abstract. *Uniqueness* requirements aim at avoiding redundancy in the summary in order to achieve maximal efficiency. *Structural* requirements deal with the presentation of different regions within the result. Finally, *navigability* requirements concern to the source video navigation options.

The next subsections contain the lists of requirements for each category that our video summarization system should fulfill.
3.2.1 Priority requirements

Priority requirements indicate which content should be preferably included in the summary to convey as much relevant information as possible in each region of the resulting image.

**Requirement P.1 People and main characters**
The system has to center on people as the most relevant content in the source video. Viewers are naturally interested in seeing the characters that are part of the video; therefore, frames showing people should be preferred for being included in the summary result.

**Requirement P.2 Fast understanding**
Although an image can contain storyline information, foremost it has to contain frames with widely known relevant objects. Such information allows users to rapidly grasp the content of the original video.

**Requirement P.3 Visual variability**
Including different video scenes in the summary will allow our system to be more efficient. Furthermore, content and scene variability will help to maximize the understanding of the abstraction employed in the summarization process.

3.2.2 Uniqueness requirements

A summary should provide unique, non-redundant information to be efficient. Uniqueness requirements aim to penalize redundancies in the content.

**Requirement U.1 Non-repetition**
An object map should not contain any repetition of a scene of the original video. This means that we have to include as much different information as possible by splitting the video scenes correctly.

**Requirement U.2 Visual uniqueness**
Representing different content maximizes the efficiency of the video summary by minimizing redundancy in the visual domain. This means that visually, the objects included in an object map should be as different from each other as possible.

**Requirement U.3 Characters uniqueness**
The object map has to avoid redundancy when representing characters. Frames or mapping regions showing main characters of a video should not be repeated.
3.2.3 Structural requirements

Structural requirements provide rules that constrain the content presentation within the video summary.

**Requirement S.1 Main characters excel**
The more a character appears in the video source, the more relevance will she have in the summary. This means that larger regions will represent the more important content.

**Requirement S.2 Style**
Resized frames or regions of interest should not be distorted. This means that the chosen representative of each relevant object must be selected as the one with the least distortion after processing. For instance, a large face will be less distorted than a small one.

3.2.4 Navigability requirements

The navigability requirements aim at allowing users to quickly navigate through the video content.

**Requirement N.1 Region boundaries**
Users must understand where the boundaries between different region representations are. This allow them to rapidly realize where they can browse to different content.

**Requirements N.2 Metadata supplement**
The system may be complemented with textual metadata. It should facilitate this task by creating a textual description of the mapping structure.

3.3 Overview and priorities

In a real system, the implementation of each requirement has a cost in terms of processing power, memory consumption and time required for computation. It is part of the design to prioritize the requirements for generating practically optimal result.
Time consuming computation is not always a priority because in video summarization we talk about an offline service. This means that the video source can be introduced in the system at any time to be processed and the application will, eventually, generate an object map representation.

Moreover, a distinction can be made between requirements that must be fulfilled and requirements in which the degree of fulfillment influences the quality of the final result without invalidating it in case of incomplete fulfillment. We assign the highest priority score, 1, to the ones that must be fulfilled while all the other cases receive scores 2 and 3. These criteria are further discussed in Chapters 4 and 5.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>P.1 People and main characters</td>
<td>1</td>
</tr>
<tr>
<td>U.1 Non-repetition</td>
<td>1</td>
</tr>
<tr>
<td>S.1 Main characters excel</td>
<td>1</td>
</tr>
<tr>
<td>P.3 Visual variability</td>
<td>2</td>
</tr>
<tr>
<td>U.2 Visual uniqueness</td>
<td>2</td>
</tr>
<tr>
<td>U.3 Characters uniqueness</td>
<td>2</td>
</tr>
<tr>
<td>S.2 Style</td>
<td>2</td>
</tr>
<tr>
<td>N.1 Region boundaries</td>
<td>2</td>
</tr>
<tr>
<td>P.2 Fast understanding</td>
<td>3</td>
</tr>
<tr>
<td>N.2 Metadata supplement</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 1 Requirements overview and priorities
Design and Implementation

After having defined the requirements that our content-based video summary should fulfill in Chapter 3, we specify the approach design. Furthermore, we describe the implementation of the elements, constraints and functions that were discussed in Chapters 2 and 3.

This chapter is structured as follows: Section 4.1 provides an overview over our solution that is further explained in subsequent sections. Section 4.7 focuses on the development environment and all the steps necessary to use the application.

4.1 Overview

Our approach to solve the video summarization generation problem consists of three main steps: a preparation step, a content selection step, and a composition...
step. Each step may also be divided into different architecture blocks (see Fig. 22). Each block aims at solving various requirements described in Chapter 3 shows an overview of the requirements and architecture blocks.

<table>
<thead>
<tr>
<th></th>
<th>Shot segmentation</th>
<th>Face detection</th>
<th>Face clustering</th>
<th>Object detection</th>
<th>Object map composing</th>
</tr>
</thead>
<tbody>
<tr>
<td>P.1 People and main characters</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>U.1 Non-repetition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S.1 Main characters excel</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>P.3 Visual variability</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>U.2 Visual uniqueness</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>U.3 Characters uniqueness</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S.2 Style</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N.1 Region boundaries</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P.2 Fast understanding</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>N.2 Metadata supplement</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 2 Linking of requirements and architecture blocks

In the preparation step, the input video is sampled uniformly. Then, shot boundary detection identifies scenes and selects frames to be included in the summary. In this step we aim at solving as many priority requirements as possible.

In the content selection step we analyze selected keyframes to extract relevant objects. We use object detection algorithms to locate regions of interest. Firstly, we focus on faces as the most relevant object in content summaries. Adding variability in the resulting object map is a second requirement that we want to solve at this stage. We cluster same person faces using face recognition algorithms. Then, we select the largest faces in the largest clusters as the representatives to be included in the map. Secondly, for general object detection we perform a color-based similarity algorithm and maximum matching score results in order to group similar objects and to select a representative for each detected object. In this step we aim at solving uniqueness requirements as well as structural ones.

Finally, in the compositing step, we create a visually attractive image composed of the most important content extracted in previous steps. We also want the object map to be as intuitive as possible to improve the experience of browsing through the source video. The user must know what are the different regions and periods of time he can navigate through. In this step we aim at fulfilling the navigability requirements.
In the next sections we describe the implementation of the blocks and we describe the third party software and approaches (described in detail in Chapter 2) we have used and those projects which have been tested but rejected.

4.2 Shot segmentation

Video summary generation requires a temporal segmentation of the source video. This process is named shot boundary detection. There exists significant amounts of methods in this area. First, we perform a uniform sampling of the source video. Then, we detect shot boundaries between sampled frames. We have been working with different shot detection algorithms in order to create the optimal temporal sampling of each video.

At this stage of the architecture block we aim at providing solutions to various requirements. Firstly, the visual variability (requirement P.3); we detect different scenes in order to reduce visual redundancy at the compositing stage. Secondly, shot segmentation is also related to the rapid understanding (requirement P.2). By merging several shot representations (requirement U.1) we squeeze as much information as possible in the summary.

4.2.1 Uniform sampling

First, a uniform sample extraction of the video frames is performed using the ffmpeg\(^2\) library wrapped in the JavaCV\(^3\) project, a Java Interface to OpenCV\(^4\) and other commonly used libraries in the field of Computer Vision. Used programming tools are described in detail in Section 4.7.

This extraction of frames is performed with a fixed sampling frame rate that depends on the length of the source video and its frame rate:

\[
\text{Sampling Rate} = \min\left( \frac{N_0}{L_i}, fps_i, fps \right),
\]

where \(fps_i\) represents the acquisition frame rate in which the input video was generated, \(N_0\) the total number of frames we actually want to keep in order to be

---

\(^2\) [http://www.ffmpeg.org/](http://www.ffmpeg.org/)

\(^3\) See Section 4.7 for further information about the development environment.

\(^4\) [http://opencv.org/](http://opencv.org/)
processed by our shot detection algorithm, and $L_i$ is the total number of frames of the input video. The second term is an upper bound and the first one is the downsampling rate \( \left( \frac{N_g}{L_i} \right) \) of the video sequence in the test environment. We will vary $N_g$ depending on the available JVM memory. The minimum number of frames we can obtain corresponds to a Sampling Rate of one frame per second.

### 4.2.2 Identification of shot boundaries

The Software Studies Initiative\(^5\) of the University of California in San Diego (UCSD) provides a very intuitive and simple source code in its Google repository\(^6\). We needed to modify the input data to get memory structures of the frame data and finally detect not only shots in the video, but the shot boundaries. Its method is described in detail in Chapter 2, Section 2.2.1.

The Binshtok and Greenshan source code includes three different methods for the shot boundary detection: a pixel-to-pixel method, a histogram-based method, and a third one based on the Hausdorff distance. Furthermore, it includes an additional option that combines the three methods with a neural network (NN). This is the best and most complete solution we could find and it is described in detail in Section 2.2.2.

However, since Hausdorff and the neural network-based solutions require considerable computational effort, we decided to discard them. In addition, given the interest of this application to obtain multiple views of every object, we adopted solutions with a tendency to generate over-segmentations of the shots to extract more faces (see Section 4.3) and object samples (see Section 4.5) and to generate more populated clusters (see Section 4.4). For these reasons, we decided to employ the pixel and histogram-based techniques.

Although the two solutions are incorporated, Binshtok *Cumulative Pixel-to-Pixel* technique is selected by default. It generates an over-segmentation that works quite well for the purposes of the project as we want to obtain different views of the same object or face.

We have reduced the temporal redundancy of a video as the first milestone to be accomplished in summary creation. Now, we focus on the content selection stage. Particularly, we focus on the type of object which is the most relevant in many videos: faces. In the next section we describe the design of our face detection algorithm.

\(^5\) [http://lab.softwarestudies.com/](http://lab.softwarestudies.com/)
\(^6\) [https://code.google.com/p/softwarestudies/](https://code.google.com/p/softwarestudies/)
4.3 Face detection

Face detection algorithm used in this architecture block is explained in Chapter 2, Section 2.4. We run a generic face detection engine provided by OpenCV based on Viola-Jones [30] [42] algorithm.

We focus on detecting different types of face views (frontal, right profile, and left profile). For this reason, we present the results in subsequent figures as follows: frontal face detections are painted yellow, right profile detections are painted blue, and left profile detections are painted green. Finally, removed detections by filtering are painted red.

Fig. 23 shows how the face detector typically presents two types of problems:

1. Overlapping detections
2. Extreme size detections

These problems can also be understood as false positive detections. The project includes two new blocks to solve these problems: size filtering and overlap filtering which complete the three face detections stages as shown in Fig. 24:

- Frontal face detection
- Profile face detection
- Horizontal flip of the input image to detect the opposite profile and correction of the coordinates.
- Size filtering
- Overlap filtering
Algorithm FACE DETECTION

Given collection of frames $V = \{f_1, ..., f_n\}$,
minimum detection size $s_{\text{min}}$,
frontal cascade file $F_{\text{file}}$,
profile cascade file $P_{\text{file}}$;

1: begin
2: Initialize output ROI structures
3: $F = \text{load classifier cascade}(F_{\text{file}})$
4: for $i = 1, ..., n$ do
5: $f_i \leftarrow \text{get frame } (f_i \in V)$
6: $\hat{c} \leftarrow \text{detect faces } (f_i, F)$
7: $\hat{c} \leftarrow \text{remove faces } (s_{\text{min}})$
8: if (profile detection flag) then
9: $P = \text{load classifier cascade}(P_{\text{file}})$
10: $\hat{p}_1 \leftarrow \text{detect faces } (f_i, P)$
11: $\hat{p}_1 \leftarrow \text{remove faces } (s_{\text{min}})$
12: $f_i' \leftarrow \text{flip image } (f_i)$
13: $\hat{p}_2 \leftarrow \text{detect faces } (f_i', P)$
14: $\hat{p}_2 \leftarrow \text{remove faces } (s_{\text{min}})$
15: overlap filtering ($\hat{p}_1, \hat{p}_2$)
16: overlap filtering ($\hat{c}, \hat{p}_1$)
17: overlap filtering ($\hat{c}, \hat{p}_2$)
18: end
19: release images ($f_i, f_i'$)
20: end for
21: end

Fig. 24 Face detection stages architecture

At this architecture block we aim at providing solutions to important requirements. Firstly, people (requirement P.1) are treated as the most relevant content in the video. With face detection we label regions of interest for this type of content in every single keyframe. Fast understanding requirement (P.2) is also improved because people may tell the story of a video. Finally, by keeping reference to the largest faces in keyframes removing extreme size detections, we aim at providing solution to the Style (requirement S.2) of the final object map.
In next subsections, every stage of the face detection architecture block is explained in detail by providing useful examples for their understanding. We can divide them into three stages: detection, size filtering, and overlap filtering.

### 4.3.1 Detection stage

The model files for face detection are provided by default with OpenCV, `haarcascade_frontalface_alt_tree.xml` and `haarcascade_profileface.xml`. The detection of frontal faces has proved to be reliable, with no need to retrain. However, the default model for profile faces included in OpenCV is significantly less reliable and that motivates the filtering methods. Fig. 25 shows how detectors work for each case and their reliability.

![Fig. 25 Results for frontal and profile face detections](image)

The output of this stage is a set of detected regions of interest that need to be filtered to avoid as much false positive detections as possible.

### 4.3.2 Size filtering stage

Small detections, both frontal and profile, must be removed because they are not considered good enough to be included in the output object map. This operation is performed with a size filter that compares each detected region with a predefined threshold.

Our first approach was to consider that faces size can be diverse depending on the camera shots (close, medium, long and full shots). The threshold should be adaptive for each image by estimating the mean size of frontal detections. We considered using a *median filter* that may work better. Thus, the outliers are not
involved in the size estimation. Then, those detections that are far from this size are removed.

Finally, we found that this solution is not good enough for neither the object map generation nor the face clustering architecture block. It means that small faces will be resized to be included on the mapping image, then, their quality would be poor to guarantee a high precision to build next stages. We consider that better representatives of main characters in the video could be found on other keyframes so a fixed threshold is selected to avoid small and poor quality faces (see Fig. 26).

![Fig. 26 Removed detections with size filtering](image)

### 4.3.3 Overlap filtering stage

The final stage of our face detection architecture is the overlap filtering process. Overlap detections between different classifiers are removed by using a simple algorithm that takes every pair of detections and compares their $(x,y)$ coordinates and sizes (width, height). This filter is used to remove those less reliable regions, which normally corresponds to profile detections:

1. Overlap between frontal faces (primary) and right profile faces (secondary).
2. Overlap between frontal faces (primary) and left profile faces (secondary).
3. Overlap between right profile faces and left profile faces.
4. *Design and Implementation*

The resulting image is shown in Fig. 28. In this example we observe how the number of detections has been reduced to one frontal and one profile. Even so, there are some overlapped profile detections within the sample classifier (right or left profile). The profile face classifier in OpenCV does not provide a good performance. If Fig. 29 result is used, the profile detected region sequences must be re-filtered. The adaptive size filtering process commented in Subsection 4.3.2 and the overlap filter should be used to remove the farthest one, whether bigger or smaller.

The output of the face detection algorithm is used as input to the next content selection stage, the face clustering or recognition method. We aim at recognizing different people appearing in the video frames to minimize content redundancy in the final object map.
4.4 Face clustering

The next block to be addressed in the proposed architecture is face clustering. This block must process all detected faces to decide:

- Which faces belong to the same person?
- Which faces appear more often in the video?

The expected inputs are several face samples but, experiments have shown that profile detections are much less reliable than frontal detection samples. For this reason, profile detection will not be used for the clustering process by default.

Our approach for clustering will be based on the recognition solutions already available in OpenCV. Although there are several face recognition algorithms, they are not suitable for this project because they require an initial ground truth to properly initialize a model for each of the faces to be recognized. In our project, no initial label set is available because there is no prior knowledge about who appears in the video. We will handle this limitation by adopting a model update approach over the features provided by Local Binary Patterns Histograms (LBPH), described in Section 2.4.3. Fig. 31 shows how LBPH have been used in order to achieve good results.

The main drawback for choosing this approach is that our framework cannot guarantee a training set of images from the same person. Also, our detected faces are not perfect and light and position settings cannot be guaranteed. For this reason, the face recognition block has been divided in two parts: pre-processing input frames before the feature extraction, and face labeling iterative method to properly update each created face model.

The face recognition block aims at meeting the expected accuracy to fulfill some of the requirements. Specifically, we want to provide the main solutions to priority and uniqueness requirements. Selecting larger clusters we should locate which characters are the most important in the video story (requirement P.1). Also, with different clusters we may obtain enough information to create a good result in terms of visual variability (requirement P.3) and fast understanding (requirement P.2). About uniqueness, we aim at minimizing characters redundancy (requirement U.3) within the summary. Finally, the recognized largest cluster may contain the character who appears more frequently in the video. This means we will highlight this region of interest in the object map (requirement S.1).
4. Design and Implementation

4.4 Pre-processing of face detection boxes

Performing face recognition directly on a raw image would probably turn into a low accuracy rate (around 10%). One of the most important limitations of face recognition algorithms is the sensitivity to lightning conditions. This problem may prevent the recognition of a same person if they are in a dark or bright location. In addition, the face should be in a very consistent position within detected bounding box, not including pixels coming from the background or hair.

The first step of our pre-processing is to convert RGB images to grayscale used for recognition. Secondly, the facial image is cropped in order to remove background pixels that add noise to the recognition process. For our project detections, a 20% of edge pixels are removed. Resizing the image to a preset size is the next step and, finally, histogram equalization automatically standardizes the brightness and contrast of all facial detections.

Fig. 30 shows the process chain for each facial bounding box detected in previous steps:

![Fig. 30 Pre-processing facial images for face features extraction](image)

4.4.2 Face labeling

The main challenge of the labeling problem is that the detected faces correspond to an unknown amount of characters, and that it is not known which pairs of detections belong to the same character. A solution based on OpenCV was adopted to simplify the system architecture, an algorithm which does not need to be trained with several manually annotated images.

The proposed solution is based on an iterative estimation of the face clusters by sequentially labeling each face, and using these automatic labels to retrain the same face recognition algorithm. This is a case of unsupervised learning, the most challenging scenario for machine learning.

Specifically, the face recognizer will predict a label and associated confidence with the existing trained data. This is none for the first images, one for the second... Then, it updates the model in order to generate a better prediction for the next
images. This algorithm has a weakness: if there are few labeled detections, the model cannot be properly created. As a consequence, many clusters with very few elements will be generated. In order to reduce this effect that appears mainly in the first facial images to be recognized when an early model is created, an iterative algorithm has been defined (see Fig. 31) and described in detail below:

1. The first face is used to initialize the face recognizer model
2. For every remaining non-labeled face, the recognition is performed. If the confidence value is greater than a manually preset threshold, the input face is labeled as a new one and the new model is created, otherwise, the matching model is updated.
3. When all faces have been labeled, those belonging to smaller clusters are discarded. They are considered as false positives from the face detector, or either belonging to people who do not appear often enough in the video.
4. The image with the highest confidence in each cluster is identified and considered as the representative sample for the whole cluster.
5. Representative samples of the existing cluster are used to initialize again the face recognizer model. Then, the algorithm iterates again starting from step 2 unless the whole loop has been already completed a predefined number of times. Our experiments have pointed out that four iterations may be enough.

This algorithm not only allows the removal of false positive face detection, but also achieves better recognition precision comparing facial images with more representatives in every iteration. The number of iterations to be considered must depend on the number of detected faces.

Results for four iterations are shown as an example (see Fig. 32, Fig. 33, and Fig. 34). We can observe how the number of clusters is being reduced every iteration in parallel with the increase of samples per cluster. On the first iteration in Fig. 32, clusters tend to contain few elements. Then, the second iteration shown in Fig. 33 shows how smaller clusters disappear because more representative elements have been chosen. In the last iteration (Fig. 34), the two main characters in the movie trailer appear in the largest clusters with backgrounds dark blue and orange.

Finally, the selection of a representative face for each cluster is another issue to be solved. We select the largest faces in the largest clusters as a representative facial image of the main people to composite the output object map. This selection is performed by sorting clusters and their elements by size. We will focus now on other relevant content to be included in the result using different object detection techniques.
Processed faces collection, \( F = \{ f_1, \ldots, f_n \} \)

### 4. Design and Implementation

**Fig. 31 Face clustering algorithm block diagram**
Fig. 32 First iteration of face clustering block

Fig. 33 Second iteration of face clustering block

Fig. 34 Final iteration of face clustering block
4.5 Object detection

The previous section has focused on a very specific class of object: faces. The system should be able to work with any kind of object class. The flexibility is achieved by adopting a generic object detector and allow the final user to train it with the object of interest they prefer. The last selection stage of the video summarization approach is the general object detection. The Thesis aims at providing to the user the opportunity to add in the object map the relevant content s/he wants. For this reason, multiple options have been considered.

There exist several solutions for object detectors. This project provides support for some of the state of the art implementations explained in Chapter 2, Section 2.4:

- Whether the user has a sample image of the object, SURF matching is used to search object of interest.
- If they have a trained cascade classifier, Haar-based object detector is used.
- Finally, a parts-based model can be used to search relevant content.

Then, a selection method of the best candidates is performed for each option to select those objects that better represent the actual object.

In this block we aim at providing solution to various requirements. Firstly, an adequate representation of the object helps the user to better understand the summary (requirement P.2). We also want to increase the visual variability by allowing the user to include several and different object classes that will be included (requirement P.3). Secondly, with the selection method developed for each type of detection features we try to reduce redundancy and provide solution to uniqueness requirements (U.2). Finally, by adding different object representations, we help the summary provider to complement properly related metadata (requirement N.2).

In next sections we explain in detail how to run the different included techniques as well as their required inputs.

4.5.1 Haar cascade classifiers

The Haar features-based object detection technique [30] [42] is widely used by researchers. This technique has two well differentiated stages: training and detection. The key point of this technique is that the required time of the detection stage is very low. The larger and complete cascade the training stage creates, the lower the detection of the object over an image in the detection stage will be (see Chapter 2, Section 2.4.1).
The cascade creation in training stage involves several techniques and utilities. They are briefly explained in Section 4.7.4 and enumerated below:

1. Create the object image database with positive and negative samples.
2. Use opencv_createsamples tool to create the binary format file of the positive dataset.
3. Use opencv_traincascade utility to create the cascade classifier.

When the cascade is created it can be introduced to the application as an input parameter to perform object detection over the keyframes.

![Car object detection examples](image)

Fig. 35 Car object detection examples

The output of our haar object detection algorithm is a list of sorted ROIs that can be included in the summary. This detection method fails at giving good results with different object views (see Fig. 35) as described in Chapter 2, Section2.4.1. For this reason, and to avoid users to spend time training their own cascade classifiers, we present a simpler real-time object detection algorithm based on robust features (SURF) in the next subsection.

### 4.5.2 Detection using SURF features

For those users that do not have trained cascades because the training process is long, we have developed an additional solution based on SURF features to reduce this training process to zero. This approach is described in detail in Chapter 2, Section 2.4.2.

In contrast to cascade classifiers, no additional training stage is required with this solution. The user only needs to select an example image of the object to be detected and introduce it in the application as an input parameter. This image will be named as training image. It is also very important that the training image contains only the object and must be free from any harsh lightning. In Fig. 36 we present two different examples to evaluate how important is the training image selection. The plane (left) is a generic object detection example. In contrast, the box example (right) is a very specific layout to extract interest points.
This method’s strength relies on being scale and rotation invariant, robust, fast and most importantly, its ability to work with a single training image. A short description of what the developed algorithm does is numerated below separated into two stages: descriptors extraction and matching strategy.

The descriptors extraction involves the training image. At this stage, we want to extract interest points from the image as follows (see Fig. 37):

1. Find robust features or interest points in the image as described in [31] with cvExtractSURF method from OpenCV.
2. Determine the location, size, and orientation of each feature.

Now that SURF descriptors have been extracted from the training image, we employ a matching strategy to match descriptors from every frame with the descriptors of the object and find out good matches. Note that if the selected training image does not represent properly the object, the matching process will be difficult (see airplane example in Fig. 38, left) and few matched points will be found. In contrast, the number of matched points in the box example (see Fig. 38, right) is extremely high:

1. Extract SURF descriptors of the test image with cvExtractSURF.
2. Find matching points between the training image’s descriptors and the test image’s descriptors using a laplacian filter for each one and finding its nearest neighbor. If the neighbor distance is lower than a preset threshold, the matching is positive.

---

7 Left: Pascal VOC 2012 dataset. Right: OpenCV sample image
3. With all matching points extracted, we try to detect the ROI where the object is represented in the image.

Fig. 38 SURF matching examples

The last step of the object detection process is to compare the number of matched points between all keyframes. We have developed a selection method that sorts all keyframes with the number of matching points. The more matching points the keyframe has, the more probability of object appearance it has.

The resulting sorted list is returned by the object detection class to the next architecture block, the summary image compositing but it is not always possible to locate a positive instance in the image because we may not have enough matching points. In the next subsection we will introduce briefly the deformable parts-based object detection method to counter the weaknesses the haar-based cascade and SURF methods have, improving their performance, and with scored detections retrieval.

4.5.3 Deformable parts-based object recognition

Finally, deformable parts-based object detection software has been added to the application. It is one of the most used methods today to solve multiple view object detection problems. The main algorithm is based on the detection method proposed by Felzenszwalb et al. [41] and described in detail in Chapter 2, Section 2.4.3.

The input of the detector is a single image, where the detection procedure is carried out. Depending on the target object class, a different object model is used. Such model is a computational description of an object class. It is stored in a binary file with .mat extension. Some of these model files are included in the released package: car, horse, person, bicycle, etc.
4. Design and Implementation

LibPaBOD\(^8\) is built on top of two already used libraries: OpenCV\(^9\) and MatIO\(^{10}\). Among other things, OpenCV library is used to handle images using the \textit{IplImage} structure as well as \textit{CvMat} and \textit{CvMatND} structures to perform matrix operations. MatIO library allows the software to read the object model file and load it into memory.

A simplified usage of the software is possible due to a developed Java wrapper that allows detecting objects over extracted keyframes (see Section 4.2). First, each extracted frame is saved into a temporal directory in order to be read by the binary. Secondly, the used model for the detection process is defined as an input parameter of our Java application. Finally, a detection threshold is preset in order to avoid as much less reliable detections (negative scores) as possible. By default, the threshold is set to 0.0 but it can be changed in the source code if the user does not get enough detections for a specified object.

Once the software has finished the object extraction, a .txt file is saved into the output directory. Each line of the file describes one object detected bounding box as follows: x coordinate, y coordinate, width, height, and detection score.

The highest scores obtained by the detector are those with more confidence of object appearance. Our application reads the output txt file and creates a detection structure class for each frame. In order to sort detected objects, all detections are sorted by their scores and better detections will be relevant object representations to be included in the resulting object map as explained in the next section.

---

\(^8\) \url{http://www.uco.es/~in1majim/proyectos/libpabod/}. Last visit on May, 28\textsuperscript{th} 2013
\(^9\) \url{http://opencv.willowgarage.com/wiki/}. Last visit on May, 28\textsuperscript{th} 2013
\(^{10}\) \url{http://matio.sourceforge.net/}. Last visit on May, 28\textsuperscript{th} 2013
4.6 Object map compositing

The object map compositing techniques are extremely powerful and informative when talking about summarization. There are many ways to create those images and it is challenging to predict the most informative objects and views to be used in the summary.

The rendering of the resulting map has been presented in two different ways: content segmentation and tile-based composition techniques. Regarding the content segmentation solution, its rendering has been divided in two different stages: foreground stage and background stage. We describe in detail these two options in next subsections showing examples and analyzing how well they comply with the requirements.
4. Design and Implementation

At this development stage, we focus on faces as the most relevant content to be rendered in the object map. This composite algorithm has been tested with faces only. For this reason, in this section we will refer to face maps instead of object maps.

4.6.1 Content segmentation solution

Firstly, foreground faces are selected with the output of the clustering method described in Section 4.4. They are segmented with an object segmentation java library added to the system architecture, the Interactive Natural Image Segmentation\(^\text{11}\). This segmentation approach [46] is based on a watershed algorithm and regions are labeled as foreground or background by applying an algorithm of markers propagation based on deformed graphs [45] [44]. Fig. 41 shows an example image and its generated partition:

In the project, detected frontal faces from the detection algorithm described in Section 4.3 are used to create marker images. This marker image is used to initialize the image segmentation algorithm, which will try to adjust these markers to the actual contents of the image. Our approach is to create an oval marker to indicate the location of the face. In addition, the vertical boundaries of the face bounding box are also used to create a negative labeled region for the background and extend it through the image to the opposite side of the detected face, represented as a blue rectangle (see Fig. 42, right).

---

Finally, we extract segmented face (see Fig. 43) as a result of expanding the markers through the image partition. As we have been working with movie trailers, we assume that faces look to the center of the image. If the face is placed on the left side of the image, it will probable correspond to a right-profile face; and a left-profile face if located on the right side. This assumption helps to achieve a natural rendering of the foreground since it will be used to place all representative faces into the map ordered by their position on the image. We split the width of the face map into N regions (the total of representative faces) and paint each ordered face left to right.

![Fig. 43 Segmented face](image)

Secondly, the background for the face map is selected as the keyframe representative of the longest shot in the video trailer. We adopted this solution under the assumption that, in movie trailers, the longest shot will also be very important for the summary.

These presented techniques for the foreground and background rendering were tested on a collection of 30 movie trailers. Some of these results are shown in the following face maps:

![Fig. 44 Face maps with largest shot background representation](image)

The results are quite promising but notice that several backgrounds are the credits of the movie or text frames. We changed the criterion for the background
selection: we used as background the whole frame where the largest face was detected. With this approach, new results were generated, as can be seen in Fig. 45.

These examples show that, if the face associated to the background is large, there might not be enough space to add other faces or objects in the map and we will not provide solution to visual variability requirement (P.3). Furthermore, foreground faces might overlap faces from the background frame, a situation that should be avoided because they may be important for the summary and we will not fulfill other priority requirements (P.1 and P.2). Finally, region boundaries distortion of the foreground faces cannot be attractive for the user (requirement S.2) and they may not be navigation-friendly (requirement N.1).

### 4.6.2 Tile-based composite solution

Our second approach for composite object maps tries to solve some of the problems we observed with the content segmentation solution. With tile-based composite we aim at generating a navigation-friendly object map. In addition, we want to excel properly main characters and background selection frame for a better understanding of the source video.

Tile-based structures are trendy and provide us a well structured solution based avoiding region overlaps. As we want to provide solution to a great amount of different content selection results, the developed composite technique is dynamic concerning the number of regions obtained in previous architecture blocks that can be plotted.

The first stage of the composition technique is to divide the map into a predefined number of regions following the next rules:
• The largest region will be on the top-left side of the map. It will contain the background selection frame, whether it is the largest shot frame or the most important face extracted from the clustering algorithm described in Section 4.4.

• On the top-right side of the resulting map we will plot all important faces returned by the clustering method. If the clustering returns less than four faces, we will create a single square region to paint the most representative one. We will create four smaller square regions otherwise.

• The bottom side of the map will be devoted to other important objects extracted from the object detector described in Section 4.5. If no object detection is performed, other frames containing less important faces are plotted. Furthermore, if no more faces are available, the bottom side of the map is removed.

Fig. 46 Example of tile-based map with void regions

When a map description is created, we plot all the different regions. Each painted region is labeled with its content frame number. We want to reduce content redundancy as much as possible. For this reason, we do not allow the same frame number for different regions.

Next figures show different results for movie trailers. As can be seen, different types of object maps are created according to the number of selected regions extracted from previous architecture blocks. Note that, a single video may generate more than one summarization result depending on the unsupervised face clustering stage:
With this solution approach we aim at providing better solution to navigability requirements (requirement N.1) by defining better region boundaries and helping the user browse the video from different frames. We also avoid overlapping problems; hence we achieve better understanding of the video content (requirement P.2). Main characters will be highlighted by relevance (requirement S.1) because larger regions are assigned to face detections with the largest amount of
elements in the cluster. Finally, the non-repetition requirement fulfillment (U.1) is possible labeling all added scenes by frame number to the description text file of the map and avoiding its repetition and, therefore, summary redundancy.

Next section describes the used technologies to create the proposed software as well as the programming tools and IDEs.

4.7 Environment

This section briefly describes the technologies and programming tools used for the application development. We also mention how the application has to be executed and all its dependencies in Subsection 4.7.1.

Eclipse\(^\text{12}\) is an open source development platform, tools and runtimes for building, deploying and managing software. It was originally created by IBM in 2001 and allows developing projects in Java, C, C++, Python, etc. For the application development, Eclipse has been used as an IDE.

Subversion\(^\text{13}\) is an open source Software Configuration Management tool. This version control system has been used with the Subclipse connector as a secure code backup and allowing shared versions of the project with the advisors. Each member in the Subversion server has a branch to develop their code and all branches are associated to a unique trunk that contains the shared version of the project.

Java\(^\text{14}\) is a programming language developed by Sun Microsystems which is now subsidiary of Oracle Corporation. Java is a general-purpose, concurrent, class-based, object-oriented language. One of the advantages of using Java is that its applications are compiled to a class file (byte code) that can run on any Java Virtual Machine (JVM) regardless of the computer operating system.

OpenCV\(^\text{15}\) is an Open source Computer Vision library of programming functions mainly aimed at real-time computer vision, developed by Intel, and now supported by Willow Garage\(^\text{16}\). It is free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. Written in optimized C/C++, the library can take advantage of multi-core processing. Its community and estimated number of downloads exceeds six million.

OpenCV libraries are not the only resources used for the project. We also have used two different binaries to train Haar cascade classifiers:

\(^{12}\) http://www.eclipse.org
\(^{13}\) http://subversion.apache.org/
\(^{14}\) http://java.com/
\(^{15}\) http://opencv.org/
\(^{16}\) http://www.willowgarage.com/
4. Design and Implementation

- `opencv_createsamples` is used to prepare a training dataset of positive and test samples in a format that is supported by `opencv_traincascade`. The output is a file with *.vec extension, it is a binary format which contains images.

- `opencv_traincascade` is used to train a cascade classifier and generate the *.xml cascade file.

Working with OpenCV Java interface is possible since version 2.4.4 out on March 1st, 2013. This project thesis began on October 2012, hence we use JavaCV. It provides wrappers to commonly used libraries by researchers in the field of computer vision (OpenCV, FFmpeg, OpenKinect, etc). To use JavaCV 0.3, you will need to install the following software: An implementation of Java SE 6 or 7 and OpenCV 2.4.3 library.

Furthermore, although not always required, some functionality of JavaCV and used in the project relies on FFmpeg 1.2.x.

Even so, associate Thesis code includes all required libraries used for Image Processing and Video management.

FFmpeg is a complete, cross-platform solution to record, convert and stream audio and video. It also is a free software project which most notable parts are:

- **Libavcodec**: the leading audio/video codec library.
- **Libavformat**: an audio/video container mux and demux library

The Graphical Annotation Tool (GAT) provides an interface to create ontologies and to label positive, negative and neutral instances for each ontology class. GAT originally generates an annotation file in XML and it is capable of generating an annotation to work with some OpenCV utilities described in Subsection 4.7.4.

In this Thesis, GAT has been used to create positive/negative image instances to generate OpenCV annotations. These annotations allow us to train object models and generate cascade files for object detection described in Section 4.5.

---

17 https://code.google.com/p/javacv/. Last visit on May, 28th 2013

18 http://www.ffmpeg.org/. Last visit on May, 28th 2013

19 http://upseek.upc.edu/gat/. Last visit on May, 28th 2013
4.7.1 Using the application

The Java software released is easy to use from the command line. The user does not need to install OpenCV and JavaCV libraries because they are also released with the code. All compiled libraries are saved automatically into the “user.home” directory if they do not exist\(^\text{20}\). Then, the software loads them to handle video IO and image processing structures used by JavaCV libraries linked to the project. All libraries are compiled to work with Windows 32-bit OS.

The objectMaps.jar can be run with the next command line arguments:

\[
\text{>> java -jar objectMaps.jar input_video [options]}
\]

where input_video is the path to the video to be summarized. All results are saved into the user.home/results/input_video_filename directory.

The command line arguments supported by the application are the different object detection (see Section 4.5) methods included in the project. The application automatically recognizes the input models by their file extensions:

- To use Haar-based cascade object detection, option will be the path to the cascade.xml file.
- To use Surf object detection, option will be the path to the image file. Current image formats supported are: .jpeg, .jpg, .png, .bmp, and .gif
- To use parts-based object detector, the model file must have .mat extension.

It is possible to run the application with more than one type of object detection algorithms. The resulting object map will be constructed by considering the order of the input option parameters: first options have more relevance to be plotted in the map than later ones.

\(^{20}\) The instructions for the installation and usage of the application can be found at the project’s source code page. http://sourceforge.net/p/objectmaps/
In this chapter we validate our object map summarization approach by means of a user study. The algorithm is evaluated in terms of the quality of the generated summaries.

This chapter is structured as follows. In Section 5.1 we present the hypothesis upon which the user study is based. The method we adopted is discussed in Section 5.2. Participants, test material and set-up are described in Sections 5.3, 5.4 and 5.5. Finally, the results of the test are discussed in Section 5.6.

5.1 Hypothesis

To evaluate the performance and the quality of the proposed summarization approach, the algorithm needs to be tested against control methods for generating video summaries. A simple algorithm that can be used for benchmarking is a uniform sampling algorithm. It generates a summary image by selecting uniform sampled frames of the source video with no content evaluation and composing them into a tile-based map. We expect summaries generated using the uniform sampling technique would be of considerably lower quality than summaries generated using our object map approach.
An additional control method for our evaluation is the manual selection of movie still frames from an online movie database, IMDb\textsuperscript{21}, in order to have unbiased realistic samples of relevant content appearance. These manually made samples certainly represent an upper-bound for the overall quality of the represented content of the summaries.

The question we aim at answering with this user study is whether our object map-based approach actually generates better video summaries than the uniform sampling method. Additionally we would like to know how much higher the quality of manually selected frames is with respect to the content representation of our algorithm. The general hypothesis we want to verify in this test is:

\textbf{H0.} Our object map-based approach provides a higher quality summary of a video item than the uniform keyframes sampling selection method.

What we mean with higher quality summary needs to be further specified in order to properly design the test. In relation to the requirements described in Chapter 3, we can split the generic hypothesis \textbf{H0} into four more specific ones:

\textbf{H1.} The object map generated by our approach can represent better the whole trailer and it is more informative than the uniform sampling method representation and less informative than a manual selection method.

\textbf{H2.} The content represented by our approach is more relevant and variable than the one generated by the uniform sampling method but less relevant and variable than the content of the manually selected frames.

\textbf{H3.} Our approach better represents main characters than the uniform sampling method but worse than the manual selection method.

\textbf{H4.} Our approach allows better navigation through the video than both uniform sampling method and manual selection representation.

In next section we discuss the method used to verify our generic hypothesis and the rating method we adopt in order to rank specific hypothesis we have presented.

\section*{5.2 Method}

Different methods to generate video summaries are compared (object map, uniform sampling, and manual selection). Each participant judges all summary versions and can see the movie trailer to have further information. The advantage of using this design is that a smaller number of participants are necessary. The

\textsuperscript{21} http://www.imdb.com/
disadvantage is that, because subjects see all summary versions, in evaluating a summary, a subject is influenced by having seen other versions.

The goal of the study is to obtain the performance of the proposed approach in subjective and objective terms [48] [49]. In objective terms, we analyze how the proposed approach improves the video understanding without seeing the whole source video and how it reflects the users’ perceived quality in terms of visual content presented. Visual redundancy and main characters appearance are evaluated. In subjective terms, the user study is designed to evaluate if our results can help user quickly grasp video content and navigability through it without watching the whole video. Note that it is generally very difficult for someone to understand the semantic content of a video from a single image without knowing any contexts. For this reason, we let the subjects choose whether to watch the trailer or not according to their knowledge about the evaluated video.

For this study we choose and integer score ranging from 1 (Unacceptable) to 5 (Excellent) which is used by The TRECVID Summarization Evaluation Campaign [48] [49] to rate all summary versions and hypothesis questions.

5.3 Participants

A total of 53 users were recruited by e-mailing the research groups and students of participating universities: Vienna University of Technology and Technical University of Catalonia. The online survey was also shared on the authors’ social networks (Facebook and Twitter). None of them had been involved in the development of the algorithms for the video summarization technique.

In order to control results submission, we started with a pilot study with the research groups’ participants and, after analyzing the data, we decided to open the survey to the social networks.

5.4 Test data

Table 3 reports the video items used in the test. They are chosen among the popular genres and well-known films. The source of each video trailer is the iTunes
Movie Trailers\textsuperscript{22} and different summaries (uniform sampled and object map) were created for each one.

All the summaries were created using related object detection models. For instance, \textit{Fast \& Furious} summary was created using a car model and \textit{Django Unchained} video was processed using a horse model running the included parts-based object detector (see Chapter 4, Section 4.5.3).

<table>
<thead>
<tr>
<th>Trailer ID</th>
<th>Title</th>
<th>Genre</th>
<th>Duration (min:sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The Intouchables</td>
<td>Biography, comedy, drama</td>
<td>2:18</td>
</tr>
<tr>
<td>2</td>
<td>The Matrix</td>
<td>Action, adventure, sci-Fi</td>
<td>2:20</td>
</tr>
<tr>
<td>3</td>
<td>16 Blocks</td>
<td>Action, crime, drama</td>
<td>2:23</td>
</tr>
<tr>
<td>4</td>
<td>Dark Shadows</td>
<td>Comedy, fantasy</td>
<td>2:33</td>
</tr>
<tr>
<td>5</td>
<td>The Twilight Saga: Breaking Dawn – Part 1</td>
<td>Adventure, drama, fantasy</td>
<td>2:30</td>
</tr>
<tr>
<td>6</td>
<td>Star Wars: Episode I – The Phantom Menace</td>
<td>Action, adventure, fantasy</td>
<td>2:25</td>
</tr>
<tr>
<td>7</td>
<td>The Fast and the Furious</td>
<td>Action, crime, drama</td>
<td>1:40</td>
</tr>
<tr>
<td>8</td>
<td>Mirror, Mirror</td>
<td>Adventure, comedy, drama</td>
<td>2:06</td>
</tr>
<tr>
<td>9</td>
<td>Resident Evil 5: Retribution</td>
<td>Action, horror, sci-Fi</td>
<td>2:30</td>
</tr>
<tr>
<td>10</td>
<td>50/50</td>
<td>Comedy, drama</td>
<td>2:49</td>
</tr>
<tr>
<td>11</td>
<td>The Lord of the Rings: The Fellowship of the Ring</td>
<td>Action, adventure, fantasy</td>
<td>2:47</td>
</tr>
<tr>
<td>12</td>
<td>The Dictator</td>
<td>Comedy</td>
<td>2:31</td>
</tr>
<tr>
<td>13</td>
<td>Django Unchained</td>
<td>Adventure, crime, drama</td>
<td>2:35</td>
</tr>
</tbody>
</table>

Table 3 Video items used in the user study

\textsuperscript{22} http://trailers.apple.com/
5.5 Procedure

The participants were given a web-based survey with a short introduction about the evaluation procedure. Then, they were asked to complete 13 online polls (see Fig. 49). In each of them, two summaries created with each of the considered techniques (uniform sampling and object map) were presented together with a visual representation of the manually selected frames from iMDb. Users were asked to select the representation which let them better recognize the movie. They were also asked to rate each representation, the object map effectiveness and attractiveness of the summarized trailer with an integer ranging from 1 (Unacceptable) to 5 (Excellent). Finally, the video trailer was embedded in order to help users to better recognize the source data.

People usually remember elements of the video items they see and use them in their evaluation during the test. The order in which the different summaries and the different video items are shown can therefore influence the outcome of the test. To minimize this influence, the presentation order should be as balanced as possible. The test aimed at satisfying the following constraints:

- The summary representations have to be seen first by the participant.
- Participants have to select the best representation before seeing the source video.
- The overall rating of the summaries should be done before watching the video.
- If the participant does not know anything about the evaluated video item, the trailer should be added and could be watched before answering the requirements questions of the proposed summary.
- Same genres should not be analyzed consecutively.

In order to evaluate our hypothesis, participants had to answer three particular questions and rate each presented option:

Q1. Which summary let you recognize the movie?
Q2. Does object map summary effectively represent the movie?
Q3. Is object map summary visually attractive?
Q4. Please, rate each Summary

Question Q1 aimed at directly testing hypothesis H1, question Q2 aimed at testing hypothesis H2 and H3. Finally, question Q3 aimed at testing directly hypothesis H4 and each representation rating process aimed at testing the general hypothesis H0.

---

http://www.jotform.com/
At the end of the test, they could report general remarks of the test and were asked how long it took to do the experiment. The test sessions lasted on average 14.41 minutes (median: 14, max: 30, standard deviation: 7.71).

**Fig. 49 Web-based evaluation survey**
5.6 Experimental results

The evaluation process considers the Mean Opinion Score (MOS) test, which is widely used measure of the system quality by averaging the ratings given by the users. Fig. 50 shows the global analysis of the Q4 ratings. The manual selection approach gives the best performance in terms of content selection (MOS = 3.40). Even so, the automatic object map approach achieves similar results (MOS = 3.29) and it performs significantly better than the uniform sampling method (MOS = 2.76). At this point, we state that our proposed approach is considered a good summary of the source video.

![MOS](image)

**Fig. 50 Global rating of the summaries**

![MOS for video](image)

**Fig. 51 Individual rating of the summaries**

We also present the individual results (see Fig. 51) for each evaluated trailer. Table 3 shows the identifiers for each video. We observe how our approach receive a “Fair” rate in trailer 1 (object map MOS = 2.75). This result was obtained by running car object detection with parts-based algorithm (Fig. 52). This solution presents content redundancy presenting two tiles with a similar scene of car detection. This makes the summary slightly worse than the other options, as can be seen in Fig. 53.
In contrast, results for trailer 7 shows how our approach (object map MOS = 3.5) summarizes considerably better the video content with a similar set up (see Fig. 54). That is using a deformable parts-based object detector using a car model. In this example we observe the importance of cars for this particular movie trailer and object map obtains nearly a “very good” rate. Furthermore, the rating for other solutions, like trailer 6, is similar for every summary option. That means that it has very specific content or well-known characters. Specifically, trailer 6 corresponds to a Star Wars movie. Thus, it has very particular characterization and clothing that allows users to rapidly select a “good” rating.

We conclude that a good summary can be obtained whether the object detector model selection is accurate. It is very important to select a model that is able to summarize relevant objects for each video in order to obtain a rich summary.
In addition to the MOS analysis, the representativeness of the summaries is assessed through a user recognition rate of the related movie, \textbf{Q1}. Fig. 55 shows the average recognition score for each technique with the following interpretation:

(a) uniform sampling  
(b) uniform sampling + object map  
(c) uniform sampling + object map + manual selection

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{recognition_rate.png}
\caption{Recognition rate}
\end{figure}

In Fig. 55 we observe that 36.78% of participants recognized the movie seeing only the uniform sampling solution. Using object maps, the amount of participants that recognized the movie were 74.76%. Finally, participants saw the manual selection option and 88.70% of users could recognize the movie. The 100% is not achieved with (c) because some users did not recognize some movies.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{individual_recognition_rate.png}
\caption{Individual recognition rate}
\end{figure}

Regarding movie recognition, we present individual results in Fig. 56. Some results are closely related to the MOS value obtained for each video. For instance, a less reliable object map summary (trailer 1) may result into a bad recognition rate. The recognition relevance of the trailer 1 object map summary is significantly smaller than other trailers. The same result is obtained with trailer 4 and trailer 9 but with different reason. The reason why uniform sampling achieves high recognition rate is
because the trailer title is shown in the summary (see Fig. 57 and Fig. 58) and they can be easily recognized.

Finally, an average measure of the attractiveness (Att) and effectiveness (Eff) of object maps summaries were asked with an integer ranging from 1 (Unacceptable) to 5 (Excellent), Q2 and Q3. The global results are presented in Fig. 59 while individual results are shown in Fig. 60.

**Average acceptance rate**

![Average acceptance rate chart](image)

**Fig. 59 Global acceptance rate**
5. Evaluation

Object maps are aesthetically valued by users between “good” and “very good” in results shown in Fig. 59 (Att = 3.34). Furthermore, their effectiveness is graded as slightly better than “good” (Eff = 3.11). These results show that our approach can effectively generate rich content summaries. Furthermore, they are attractive for them, so they can be used to represent videos without requiring user attention during some time seeing a video preview. Object maps do not need any interaction to rapidly grasp the image content, it is a static and non-stressful content result environment.

<table>
<thead>
<tr>
<th>Trailer ID</th>
<th>Object detector</th>
<th>Duration (min:sec)</th>
<th>1sec processing time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>H</td>
<td>2:18</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>N</td>
<td>2:20</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>P</td>
<td>2:23</td>
<td>88.6</td>
</tr>
<tr>
<td>4</td>
<td>N</td>
<td>2:33</td>
<td>0.8</td>
</tr>
<tr>
<td>5</td>
<td>N</td>
<td>2:30</td>
<td>0.7</td>
</tr>
<tr>
<td>6</td>
<td>N</td>
<td>2:25</td>
<td>0.7</td>
</tr>
<tr>
<td>7</td>
<td>P</td>
<td>1:40</td>
<td>70.1</td>
</tr>
<tr>
<td>8</td>
<td>N</td>
<td>2:06</td>
<td>1.3</td>
</tr>
<tr>
<td>9</td>
<td>H</td>
<td>2:30</td>
<td>2.2</td>
</tr>
<tr>
<td>10</td>
<td>N</td>
<td>2:49</td>
<td>0.8</td>
</tr>
<tr>
<td>11</td>
<td>P</td>
<td>2:47</td>
<td>28.2</td>
</tr>
<tr>
<td>12</td>
<td>N</td>
<td>2:31</td>
<td>1.5</td>
</tr>
<tr>
<td>13</td>
<td>P</td>
<td>2:35</td>
<td>55.6</td>
</tr>
</tbody>
</table>

Table 4 Object map summary setup

We have been talking about different summaries created using different object detectors. It is also important to measure the computational effort for the generation of these summaries. The processing time in the summary creation process is mainly related to the used object detection technique. If parts-based object detector is used, the processing time is incredibly higher (see Table 4).
so, the results are better as it has been demonstrated in the user study and the performance of the object detector is highly improved than using other options as we can obtain different views of the same object class. With Haar cascades, we can obtain different objects to be included in the summary by slightly incrementing the processing time. Table 4 shows the object detector algorithm used for each trailer, the duration of the trailer, and a summary processing time value related to one second of the source video. The object detectors have the following interpretation: N (None), H (Haar cascades), P (Deformable parts-based), S (SURF matching).
Conclusions

This Thesis has been developed in compliance with the requirements stated by the Technische Universität Wien (TU Wien) and Universitat Politècnica de Catalunya (UPC). We aim at providing a solution for video content summarization using relevant objects, analyzing the video and helping users to understand a video content item in a fast and visual way. We use various Computer Vision techniques for visual content analysis for creating an open source algorithmic approach to generate object map-based summaries.

We have elicited user needs with respect to video summarization by analyzing related literature on video summarization and relevant content extraction. The proposed solution takes into account several requirements to allow fast and convenient content selection to be introduced in the summary. These requirements can be divided into four categories: priority, uniqueness, structural, and navigability. Priority requirements indicate which content should be preferably included in the summary to convey as much relevant information as possible. Uniqueness requirements state that a summary should provide unique, non-redundant information to be efficient. Structural requirements provide rules that constrain the content presentation within the video summary. Finally, navigability requirements deal with the rapid selection of important scenes.

Based on these requirements, we have created an open source Java software specialized for the generation of object maps. Our solution approach is based on three main steps (see Figure. 22 in Chapter 4, Section 4.1): preparation, content selection, and composition. In the preparation step the video is sampled uniformly. Then, shot boundary detection algorithm selects keyframes to be analyzed in the next steps. In the content selection step we analyze selected keyframes to extract relevant objects and recognize them. We use Haar cascade-based face detection to extract faces of each keyframes. Then, a recognition process based on Local Binary Patterns Histogram features is used to cluster those detections that belong to the same person and select a representative face of the most important clusters to composite the output object map. We also provide solution to general content detection that can be customized by the user to include different types of content into the summary. User can use different solutions such as Haar cascade classifiers,
SURF matching, and Deformable parts-based object detection to extract relevant content. Finally, in composite step we create a visually attractive tile-based image composed by the most important content extracted in previous steps. The object map aim at being as intuitive as possible to improve the browsing experience through the video.

Our approach was evaluated with a user study in which we compared our object map solutions to human-made still frame selection and to uniformly sampled frames summaries. The results have shown that presented approach is able to properly include relevant content in a visually attractive and effective way. The computational effort to create the object maps is mainly related to the used object detection technique. If parts-based object detector is used, we process each video second in 60 seconds. In contrast, using Haar-based solution or SURF matching, the computational effort decreases considerably to 2 seconds per video second.

Finally, the Thesis provides answers to the four research questions presented in Chapter 1, Section 1.1. The first question is related to the quality of a video summary into a single image. The results have shown that object maps representations can effectively summarize the video content in a static image. The second question is related to the composing method of the different extracted faces or objects from the source video. We have tested two different representations (see Chapter 4, Section 4.6): segmentation-based and tile-based. The second one provides a more attractive solution for users because they contain more background information and, thus, users can effectively recognize presented scenes. The third question states which content should be extracted from the source video in order to understand it. We decided to develop a solution that provides flexibility to the user, as the system can be tuned with one or multiple object classes, as long as a valid model has been trained for that purpose.

This kind of video summarization systems can be widely used to manage large video collections. For instance, User Generated Content sites may summarize its videos using proposed approach to increase the content accessibility of the viewers. Broadcasting Corporations may also be interested of using our approach. Companies related to audiovisual production have to deal with an incredible amount of data which may be reviewed and indexed by documentalists. We can facilitate their work by presenting a single image and let them grasp rapidly the most relevant content.

The most remarkable contribution of this project has been developing and testing an open source software that is able to create rich summaries with customized content into a single image:

- We have developed software that can summarize videos using content-based visual analysis. The generic approach of the proposed system allows users to select which object classes are relevant and, then, allows them to easily
introduce selected models into the software to finally create a custom summary of a video item.

- The developed tile-based object map compositing algorithm allows users to rapidly grasp video content and navigate through the video.
- The Open Source software allows users to change the default algorithms used during the summary creation process. Sharing the code is the best option to adapt the software to the user needs. Our approach is publicly available at sourceforge site\textsuperscript{24}.

\textsuperscript{24} http://sourceforge.net/p/objectmaps
Future work

At this point, we identify eight main directions of research along which the presented work could be taken further:

- Face clustering.
- Audio content analysis and understanding.
- Video sequence analysis.
- Content presentation.
- Movie semantics retrieval.
- OCR stage.
- Object detector selection.
- Social media.

The main weakness of the presented work is the face clustering method. As we do not have any information about the number of clusters to be created to group all characters, neither the necessary ground truth to perform most used recognition methods, our clustering method may fail at being accurate and stable. Different runs over the same video item can generate different clusters and, therefore, different representatives are selected creating different summaries. This process allows users to re-run the software in order to get different summary representations and select the best one.

Some changes can be done in order to solve this problem. Running the proposed face clustering algorithm several times may result into recognize good clusters: those faces that are clustered in a same model may be correctly grouped. Furthermore, other clustering methods can be used to solve this problem. Affinity Propagation [10] can be used to generate stable clusters and face representatives. Liyan Zhang et al. [50] propose a unified framework that automatically learns adaptive rules to integrate heterogeneous contextual information (people co-occurrence, human attributes, clothing...) along with facial features to improve unsupervised face clustering recall while maintaining very high precision.

The basic elements of our summarization technique are visual keyframes. We do not perform any audio analysis of the video in order to discover speech, special effects (such as explosions, shots, etc.) that are semantically meaningful for human beings. This analysis can improve considerably the content selection for the summary
Future work

due to they always grab human attention and help to better understand specific video items such as movies, trailers, or TV shows.

In order to reduce the false positive detections of the different proposed object detection techniques, a sequential analysis over continuous frames may be performed. It is also effective to include motion and spatial activity analysis to define best candidates to represent a relevant semantic class.

The compositing stage of a summary is one of the most important parts of the architecture. A good representation can make the difference between good and bad summaries. The amount of source frame pixels represented in the summary is a variable that has to be taken into account if we want to create rich content-based summaries. For instance, it is not the same to represent a face or a car; user may also want to see the environment that comes with a character and does not need to have a big face to recognize the actor/actress appearing in the map. In contrast, we need to see a car brand to effectively recognize the car model. The type of object defines the importance of its details. Furthermore, the composition algorithm may benefit by some sort of visual similarity assessment that would guarantee diversity in the results.

Based on the results ratings presented in Fig. 51, it is important to have a prior knowledge about the relevant semantics of the movie. This could be achieved by searching on the web for textual descriptions about the movie. This would provide names of the main actors/actresses and objects. With these words, an online image search could be run to retrieve images of that object and people. These images could be used to train classifiers and run the detectors.

Regarding the movie recognition results (see Fig. 56), some random results have a high recognition score because the movie title appears. We conclude that an OCR stage in the video analysis could improve Object Maps results.

Now, users must specify which object detector they are using for each input video. This should also be automatically solved to improve the Object Maps experience. With some movie semantics knowledge, the software should be able to select which kind of detector could be used in order to generate the summary.

Object Maps could be used in social media. Social networks need some solutions to properly present video content. Today, a keyframe representative is shown in most of them. In addition, social media videos are quite different from others: they have few scene changes and poor quality. In order to use the presented approach with this kind of videos, we may want to analyze the video in a more generic way, presenting a global perspective of the video.
Bibliography


Bibliography


This appendix shows object maps used for the web-based survey described in Chapter 6. They are generated using different types of object detection models such as car cascades for Haar-based cascade classifiers, car (see Fig. 61, Fig. 67, and Fig. 68) and horse (see Fig. 66) models for the deformable parts object detection software.

Fig. 61 16 Blocks

http://www.youtube.com/watch?v=9B1bXeNUWGc
Fig. 62 50/50

http://www.youtube.com/watch?v=mMaJET7mD0M

Fig. 63 The Twilight saga - Breaking Dawn part 1

http://www.youtube.com/watch?v=PQNLfo-SOR4

Fig. 64 Dark Shadows

http://www.youtube.com/watch?v=wpWvkFlyl4M
A. Test data

Fig. 65 The dictator

Fig. 66 Django Unchained

Fig. 67 The fast and the furious

29 http://www.youtube.com/watch?v=DS2lURW4JSI
30 http://www.youtube.com/watch?v=eUdM9vrCbow
31 http://www.youtube.com/watch?v=2TAOizOnNPo
Fig. 68 The Intouchables

http://www.youtube.com/watch?v=34WIbmXkewU

Fig. 69 The Lord of the Ring - The Fellowship of the Ring

http://www.youtube.com/watch?v=Pki6jbSbXIY

Fig. 70 The Matrix

http://www.youtube.com/watch?v=r1GrMAqwWcl
A. Test data

Fig. 71 Mirror, Mirror\(^{35}\)

Fig. 72 Resident Evil 5: Retribution\(^{36}\)

Fig. 73 Star Wars: Episode I - The Phantom Menace\(^{37}\)

\(^{35}\) http://www.youtube.com/watch?v=YgbH05rQx1s

\(^{36}\) http://www.youtube.com/watch?v=HYuxE3YetQo

\(^{37}\) http://www.youtube.com/watch?v=1dWA9DwDQpM
This appendix makes an overall analysis of the Pascal Visual Object Classes Challenge (VOC) 2012 data that has been used to generate training datasets for the OpenCV object training process. The next sections will explain in detail how the database is structured and the developed Java code used to extract training data. Next, we describe OpenCV utilities used during the training process to generate the desired cascade.

B.1 Pascal VOC 2012 structure

The main goal of this challenge is to recognize objects from a number of visual object classes in realistic scenes. It is fundamentally a supervised learning problem in that a training set of labeled images is provided. The twenty object classes that have been selected are:

- **Person**: person.
- **Animal**: bird, cat, cow, dog, horse, sheep.
- **Vehicle**: aeroplane, bicycle, boat, bus, car, motorbike, train.
- **Indoor**: bottle, chair, dining table, potted plant, sofa, tv/monitor.
There are three main object recognition competitions: classification, detection, and segmentation, a competition on action classification, and a competition on large scale recognition run by ImageNet\(^{38}\).

The training data provided consists of a set of images; each image has an annotation file giving a bounding box and object class label for each object in one of the twenty classes present in the image. Note that multiple objects from multiple classes may be present in the same image.

In this Thesis, we have used the development kit. After untarring it, the resulting directory structure is:

- **VOC2012/ImageSets/Main** directory contains text files specifying lists of images for the main classification/detection tasks. The files `train.txt`, `val.txt`, `trainval.txt` list the image identifiers for the corresponding image sets (training, validation, training + validation). Each line of the file contains a single image identifier and whether the current object exist (1) or not (-1).
- **VOC2012/JPEGImages** directory contains all source images.
- **VOC2012/Annotations** directory contains all the annotation files giving a bounding box and object class label for each object appearing in the image.

### B.2 Dataset generation using Java

The delivered Java code contains two classes that help users in generating specific datasets of a specific object to generate haar classifiers cascades using OpenCV tools.

The class `src.objectDetection.utils.PascalTrainingDataGenerator.java` class generates to the output directory two subdirectories containing both grayscale positive and negative labeled images. It also generates positive.txt and negative.txt files that contain a list of image identifiers prepared to be read by `opencv_createsamples` and `opencv_traincascade` tools.

\(^{38}\) [http://www.image-net.org](http://www.image-net.org)
This class is easy to use: the input variables are the path to directories containing VOC images, text files, annotation files, output directory to save the results, and the name of the class to be extracted.

B.3 OpenCV tools to generate Haar cascades

For training we need a set of samples. PascalTrainingDataGenerator.java class can generate positive and negative samples easily. Sets of negative samples obtained by the Java class are already prepared to support cascade training process, whereas positive samples must be created using opencv_createsamples utility.

Negative samples are taken from arbitrary images. These images must not contain detected objects. They are enumerated in a special file, a text file in which each line contains an image filename (relative to the directory of the description file) of negative sample images. Note that negative samples are also called background samples. Described images may be of different sizes but each image should be (but not necessarily) larger than a training window size, because these images are used to subsample negative image to the training size. An example of description file:

Directory structure:

```
/negative
  img1.jpg
  img2.jpg
negative.txt
```

File negative.txt:

```
negative/img1.jpg
negative/img2.jpg
```

Positive samples are created by opencv_createsamples utility. Note that you could need a large dataset of positive samples before you give it to the mentioned utility, because it only applies perspective transformation. For instance, you only need one positive sample for absolutely rigid object like a logo, but you definitely need hundreds and even thousands of positive samples for faces, cars, etc.

The format of the positive description file is as follows:

```
[filename] [# of objects] [[x y width height] ... 2nd object] ...
```

Where (x, y) is the left-upper corner of the object bounding box and its width and height.
The `opencv_createsamples` utility crops regions specified and resize these images and convert into .vec format. Using description file obtained by the PascalDataGenerator.java class, the used command line arguments for this utility are:

- `-info`: Description file of marked up samples.
- `-vec`: Name of the output file containing the generated samples.
- `-w`: Width (in pixels) of the output samples.
- `-h`: Height (in pixels) of the output samples.

Note that for training, it does not matter how vec-files with positive samples are generated, but `opencv_createsamples` utility is the only one way to collect/create a vector file of positive samples, provided by OpenCV.

The next step is the training of classifier. There exist two solutions to train cascades: `opencv_traincascade` (the newer) and `opencv_haartraining`. In this section only the newer tool will be described further. Most frequently used command line arguments of `opencv_traincascade` are:

- `-data`: Where the trained classifier should be stored.
- `-vec`: Vec-file with positive samples.
- `-bg`: Background description file of negative samples.
- `-numPos`: Number of positive samples used in each classifier stage.
- `-numNeg`: Number of negative samples used in training for every classifier stage.
- `-numStages`: Number of cascade stages to be trained.
- `-w`: Sample width. It has to be the same value used in the `createsamples` utility.
- `-h`: Sample height. It has to be the same value used in the `createsamples` utility.
- `-baseFormatSave`: This argument is actual in case of Haar-like features. If it is specified, the cascade will be saved in the old format.
- `-minHitRate`: Minimal desired hit rate for each stage of the classifier. Overall hit rate may be estimated as \(\text{min\_hit\_rate}^\text{number\_stages}\).
- `-maxFalseAlarmRate`: Maxima desired false alarm rate for each stage of the classifier. Overall false alarm rate may be estimated as \(\text{max\_false\_alarm\_rate}^\text{number\_stages}\).

After the `opencv_traincascade` application has finished its work, the trained cascade will be saved in cascade.xml file in the folder, which was passed as `-data` parameter. Other files in this folder are created for the case of interrupted training, so you may delete them after completion of training.
The training process is finished and you can test your cascade classifier. We share a simple Java class that measures the performance of the generated cascade using Pascal VOC validation data as described in the next section.

### B.4 Cascade evaluation using Java

The final step of the object detection training process is the performance evaluation of the generated classifier cascade. Again, the provided Java code contains a class that helps users to evaluate it easily.

`src.objectDetection.utils.PascalDetectionEvaluation.java` class uses the VOC development kit evaluation data to measure the generated cascade performance. It detects selected class objects in the images listed in the evaluation description file using the haar object detection algorithm described in Chapter 4, Section 4.5.1. The input variables to use this class are the object class name, the cascade to be evaluated, the directory containing the evaluation description files, the directory containing the source images, and the output directory to store the results.

The results are presented as follows:

- Two subdirectories are created with both positive and negative detections. Positive detections bounding boxes are painted and saved into the positive subdirectory.
- A text file containing the results of the evaluation: The total of true positive, false positive, true negatives, false negatives, and the evaluation measures precision and recall.