Content-based video summarization to Object Maps

by Manuel Martos Asensio

directed by Horst Eidenberger and Xavier Giró-i-Nieto

Technische Universität Wien (TUWien)

Universitat Politècnica de Catalunya (UPC)

2012-2013

Contents

List of Figures

List of Tables

Agraïments

Voldria aprofitar aquesta oportunitat per expressar la meva gratitut 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 familia 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 in 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, e.g. search command 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 descriptions 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 this content properly in a fast and visual 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 with 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 it.

Selecting the main content for the summary is performed 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.

Fig. 1 Video summary example of relevant faces

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

1. How good can be a single image representation of video content?

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 suggested:

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 in order to create a good representation. We will analyze user attention models approaches [2] to detect where would we 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 users' point of view?

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, relevance... of the video. This data does not always give enough information to the user 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 group different keyframes information, 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 to rapidly navigate through the original video giving him the opportunity to select which parts are important. 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 aid to include hyperlinks of 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 where each object was extracted from without using sliding bars or other techniques, only with a simple click.

Furthermore, the visual representation would complete 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 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 dramatically reduced.

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 skimming. We then describe others 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 solution 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 self-trained object detection methods. Our solution approach is validated in Chapter 5 by means of a user study, and in Chapter 6 and 7 we discuss our conclusions and future work.

2

Video Summarization: Related work

In this chapter we describe the video summarization techniques to achieve new levels of understanding. We begin on Section 2.1 with an explanation of existing types of video summarization techniques. Then, in subsequent sections we will 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 our final architecture 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 Video summarization. Definitions

Video summarization engages in providing concise and informative video summaries in order to help people browsing and managing video files 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. They can be classified into two types: *Overview* and *Highlight*.

In the classic case of movie trailers, the user is usually unaware about 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, the user wants to see the most important events in the video (goals, news headlines) according to their interests. This kind 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 *static storyboard*, is a small collection of salient images or a single one extracted or generated from the underlying video source. 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 uniform sampling 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 difference 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 will review different technologies used to construct the commented summaries as well as techniques to temporally segment the source video and extracting relevant object content (faces, cars, etc).

2.2 Shot segmentation

Temporal redundancy is a very important issue to solve 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*.

Fig. 2 Shot boundary detection example

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 to a 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 positive 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 images. Another technique is to calculate the difference of all channels between the histograms in the two images and take the maximum to the summation in order to detect intense changes in one channel. Finally, a variation of the last mentioned technique is to weight the importance of each color channel.

A method that uses Hausdorff approximation to determine the outliers is used in [13]. 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 correlating edge exists in the successive image. If the sum of non correlated edges is greater than some threshold, a shot boundary is declared.

Fig. 3 Shot detection example using Hausdorff distance method

[13] also presents a combination of all the commented methods by building a *Neural Network* (NN) which inputs are the outputs of the different commented 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 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 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:

- First approaches compare pixel intensity and image histogram to decide whether two frames belong to the same shot.
- Later approaches include edge evaluation comparison between frames using Hausdorff distance.

- Learning processes using NN are also used to adapt the shot detection to the source video regardless preset 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 chosen approaches used in the Thesis, the development of which development is explained in Chapter 4, Section 4.2.

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 between frames. If the pixel change is greater than some threshold and its cumulative sum is greater than the region threshold for the frame threshold number of regions in the frame, then it triggers the shot boundary detection. This technique also provides a simple frame averaging to avoid luminosity 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 Shot boundary detection using UCSD pixel regions difference).

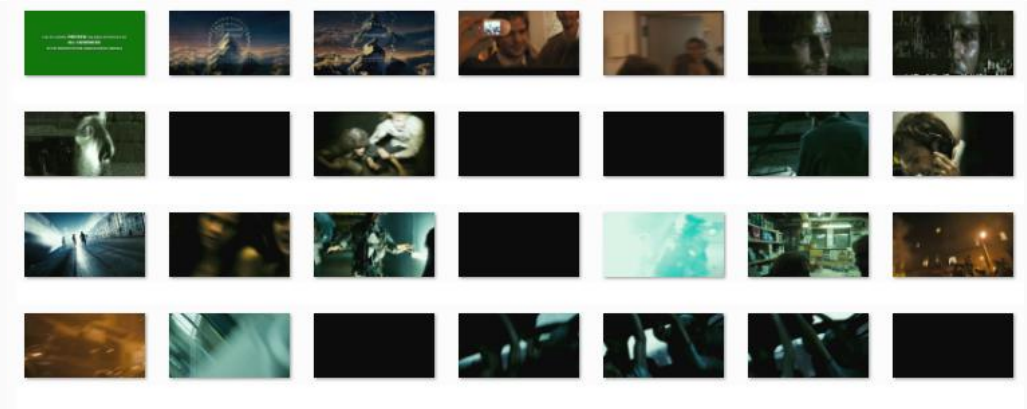

Fig. 4 Shot boundary detection using UCSD pixel regions difference

2.2.2 Course Project 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 three different methods 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. 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 generate over segmentations of the videos due to changes in luminosity or points of view as shown in Fig. 5.

Fig. 5 False shot detection useful for the project. Frontal and side views

There are many types of pixel comparisons provided by 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 formula (1).

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

  $I(t,i,j)$ represents the intensity value of pixel (i,j) at time frame t. If the difference is bigger than some threshold (τ) value, 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 difference 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).

$$\frac{\left[\sum_{i=1}^{X}\sum_{j=1}^{Y} |\, I(t,i,j) \; - \; I(t-1,i,j)|\,\right]}{256 \cdot XY} > \tau \qquad (2)$$

The histogram-based methods compare the pixel histograms of neighboring frames to determine the shot boundaries. They introduce robustness in front of camera and object motions, but they fail into segmenting two shots whose colors are similar. Presented methods are:

- *Simple histogram:* This method calculates histogram of each color channel that form the image and the difference between the bins in each histogram of the two images using (3).

$$\frac{\sum_{C \in \{channels\}} \sum_{b=0}^{Bins} |H(t,c,b) - H(t-1,c,b)|}{|pixels| \cdot |channels| \cdot 2} > \tau \quad (3)$$

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

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

$$\frac{\left[\max_{C \in \{channels\}} \sum_{b=0}^{Bins} |H(t,c,b) - H(t-1,c,b)|\right]}{|pixels| \cdot 2} > \tau \quad (4)$$

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

$$\frac{\sum_{C \in \{channels\}} \sum_{b=0}^{Bins} \frac{w_c}{w_{mean}} |H(t,c,v) - H(t-1,c,v)|}{|pixels| \cdot |channels| \cdot 2} > \tau \quad (5)$$

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

Finally, Binshtok and Greenshpan's thesis states that the option that combines the three methods using a neural network provides the best results for the typical keyframe extraction. This is because preset input thresholds values (τ) do not play any role in the shot boundary detection. Instead, NN is adapting to any given value by propagating the errors to its weights.

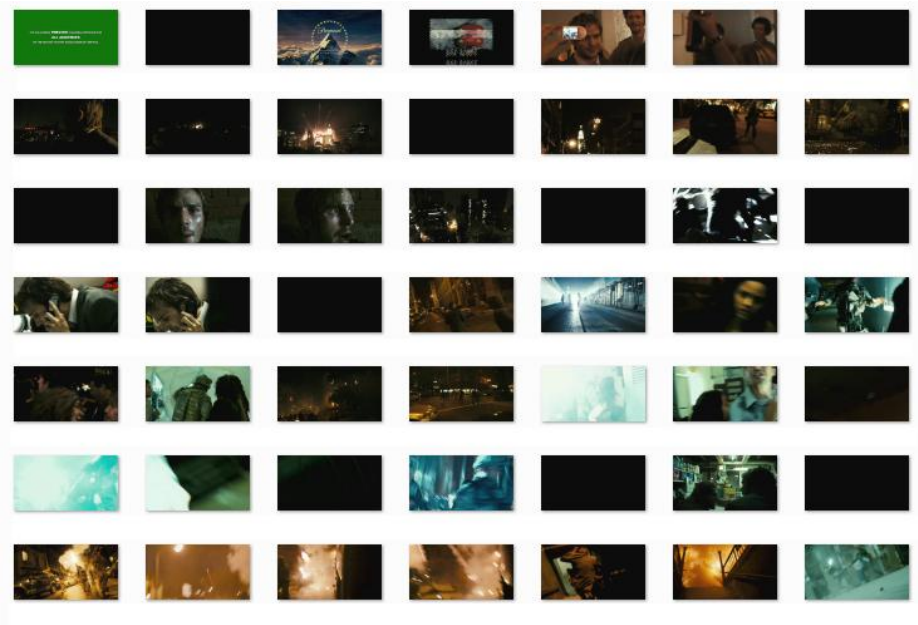

Fig. 6 Shot boundary detection using NN

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

2.3 Content selection

In this section we will analyze different approaches for the content selection included in different video summaries. We will begin on video skimming generation in Section 2.3.1 and we will continue 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 without a well-defined structure. Over the years the trend changed to include 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 video category has to be known to obtain relevant content properly and they may not be used on generic videos.

2.3.1 Dynamic video skimming

Dynamic video skimming 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 let them extract high-action parts in the video. In addition, the average color composition of the whole video is considered to include similar frames in the video skimming. 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 speed of the video when playing static video scenes and slower speed for short and dynamic video scenes. It uses audio time scale modification technology to preserve comprehensibility of speech. However, the maximum time compression depends on the speech speed. 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 *Term-Frequency – Inverse Document Frequency* technique. This text is used to create skimming version of the audio including some neighboring segments for better comprehension. Then, the image skimming is created by selecting 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. 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 explicit 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 Framework of user attention model [7]). Attention is described as a neurobiological conception that implies the concentration of mental powers upon an object or audio track. Computing attention allows them to avoid the problem of semantic understanding of the video content. Their 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 generate the summary.

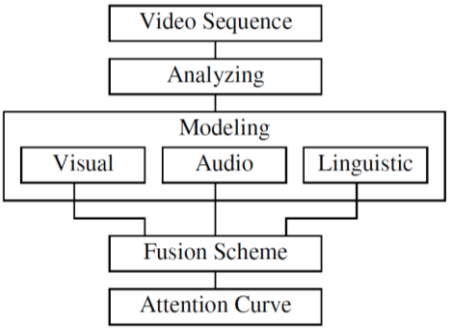

Fig. 7 Framework of user attention model [7]

This method constructs a video summary without fully semantic content understanding. However, humans do not only understand videos by perceiving these low level features. Also, the fusion scheme of the attention model parts has to be improved because it is not proved whether it is the most effective and the video structure information 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 compute 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 measure clusters can be created to reduce redundancy of the summary and thus achieve good compression ratios. With partial semantic understanding, good satisfaction and informativity measures are achieved in their experimental results.

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

- First approaches aim at providing summaries without analyzing the content. They select important scenes based on image low-level properties.
- Later approaches provide scenes based on important content. Content selection techniques involve both user attention study 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 summary

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 the video image sequence. These methods are simple and they are unable to guarantee that important content may be covered by the selected result.

Shots are an important block of the video. They represent a continuously captured sequence and shot transition detection has been suggested in various work [16] [17] [18] [19] since its visual content can be represented by some frame. These methods extract always 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 include it to 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 to background noises and motion but its performance highly depends on a 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 by 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 resulting summary.

[23] creates a mosaic image to represent the whole content of a dynamic video shot when the camera motion can be detected (pan, tilt, zoom, translate).

Although this approach is quite 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 quite complex, this algorithm tends to achieve poor performance.

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 video structure reflects how the editor chooses and arranges video shots; they are very valuable information for video summarization.

[25] provides static video summary consisting of three major procedures: keyframe extraction regarding temporal information; estimating Region 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 abstraction and navigation of the video. 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 interactively manipulate the video cube to not only understand the content but also navigate the content of interest.

[27] approach automatically extracts and visualizes movie storylines in a static image for the purposes of quick overview. *Visual Storylines* preserves the elegance of original videos with a series 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 then organizes both location and interested objects and characters (see Fig. 9). This kind of representations can be used to assist viewers to grasp video content efficiently, especially when a text synopsis is provided.

Highly condensed video summary techniques in which selected keyframes are packed and visualized using irregular shapes [28] have a common problem: due

to its highly compact form and losses of information it is nearly impossible for viewers to extract stories. Furthermore, *Visual Storylines* solves the problem of [29] by revealing the information of locations and relations between interested objects.

Fig. 9 Visual story lines example [27]

To sum up, static summary generation uses similar methods to the video skimming adding different composite techniques to correctly plot all selected information:

- First static summarization techniques are based on a single keyframe representation.
- Early approaches represent important content in a mosaic and comic-based representations that include 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 aim at providing a fast understanding of the video content and rapidly grasp the information.

For our approach purposes, we focus on the video relevant content to build our summary. Next section describes in detail briefly several content selection techniques that can be used to extract relevant content. Then, used techniques and 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.

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: *integral image, AdaBoost* machine learning and *cascade* generation that combines increasingly more complex classifiers. It is definitely well-tested, scale invariant and works fast. However, it is not rotation invariant and requires long training time.

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

There exist other visual features that have been proved to 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 *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 really bad. For this reason [33] propose, with *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 can avoid the 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, multiple rotations, 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. Cluster boosted tree classifier [37] applies a conventional *k-means* clustering algorithm to split samples when learning rates are low. They 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 [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 between different people. 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 simultaneously all subcategory classifiers 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 reducing the risk of having outliers.

A method that has become quite popular is the discriminatively trained, multiscale, deformable part model for object detection [39] [40]. As described in the related literature, detection with deformable part model can be done by considering all possible of a distinguished “root” part and, for each of those, finding the best configuration of the remaining parts. 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. They introduce *probably approximately admissible* thresholds that provide theoretical guarantees on the cascade performance and can be computed from a small sample of positive examples.

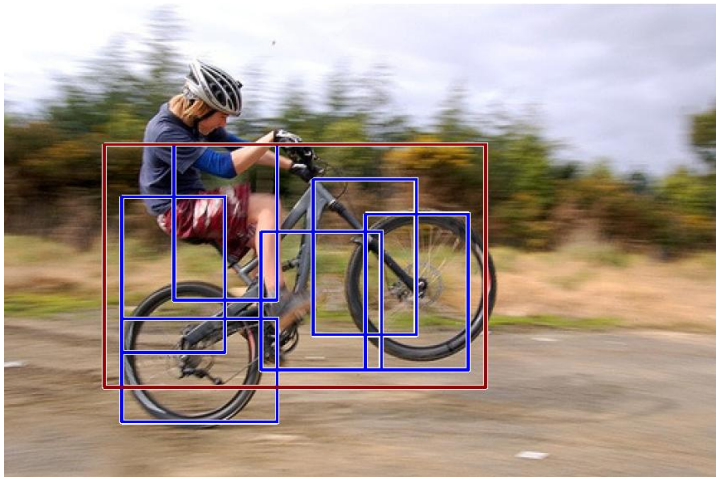

Fig. 10 Deformable part model detection [41]

To sum up, the object detection methods have evolved into different parts or views training processes to improve its performance:

- First approaches training processes do not take into account different views of the object, thus, they have not good performance 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 viewpoints are later approaches solutions 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 algorithms we have used in our Thesis to extract relevant content as well as different features we have tested.

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 image concept, *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 light one. If the difference exceeds some threshold set during the training process, the feature is said to be present.

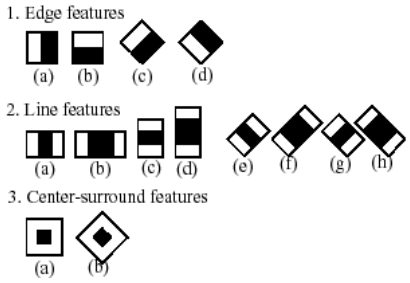

Fig. 11 Haar-like features used in OpenCV

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 capitalizes

on 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.

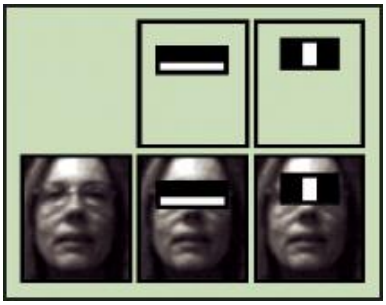

Fig. 12 First two features selected in Viola-Jones algorithm

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.

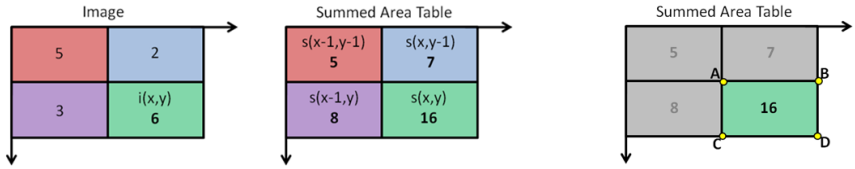

Fig. 13 Summed Area Table example

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

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x', y') \qquad (6)$$

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 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.

AdaBoost machine-learning method combines many simpler 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 approaches zero exponentially in the number of round. In their approach a variant of *AdaBoost* is used both to select small set of features and train the classifier.

The weak learning algorithm is designed to select the 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. Then, a weak classifier h_j thus consists of a feature f_j, a threshold θ_j and a parity p_j indicating the direction of the inqueality sign:

$$h_j = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

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 had 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 classifier is constructed to achieve increased detection performance. *AdaBoost* gives weights to each stage and set the order of filters in the cascade. The higher weighted filter comes first to eliminate non-face regions as soon as possible.

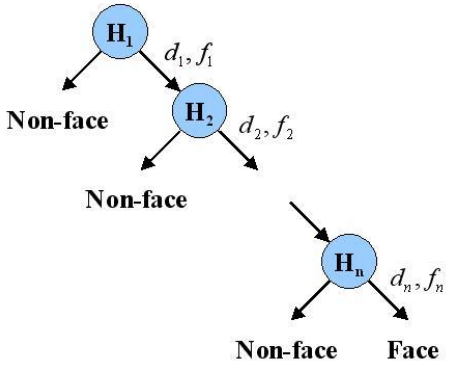

Fig. 14 Cascade classifier for face detection

A cascade is a degenerate 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 threshold yields higher detection rates and higher positive rates. Then, an excellent first stage can be constructed by reducing the threshold to minimize false negatives: it can be adjusted to detect 100% of positive object samples with a false

positive rate of 40% but it would require a significant amount of time to be processed.

The structure of the cascade reflects that, in any single image, a majority of sub-window are negative. Hence, the cascade attempts to reject as many negatives as possible at an early stage. While 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 task is more difficult than the previous ones 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 time to compute. 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 an extremely difficult 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 met.

2.4.2 Speeded-Up Robust Features

SURF [31] is a scale and rotation-invariant interest point detector and descriptor. It approximates or even outperforms previously proposed schemes with respect to repeatability, distinctiveness, and robustness, yet 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 Hessian 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 σ is defined as follows:

$$H(X,\sigma) = \begin{bmatrix} L_{xx}(X,\sigma) & L_{xy}(X,\sigma) \\ L_{xy}(X,\sigma) & L_{yy}(X,\sigma) \end{bmatrix} \quad (8)$$

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.

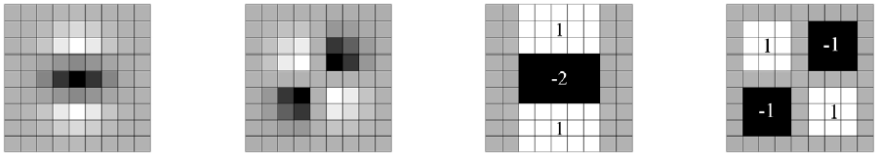

Fig. 15 Discrete Gaussian second derivative box filters

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 to the selected orientation, and finally they extract the SURF descriptor from it.

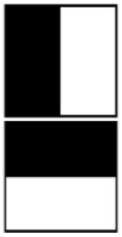

Fig. 16 Haar-wavelet in x and y directions

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 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 observe 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.

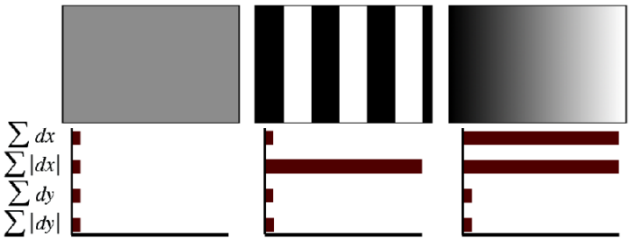

Fig. 17 SURF descriptor performance in different image intensity patterns

2.4.3 Local Binary Pattern Histograms

Unlike *Eigenfaces* and *Fisherfaces*, LBPH extract 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 coded with 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 code the relation with 0; code with 1 otherwise (see Fig. 18).

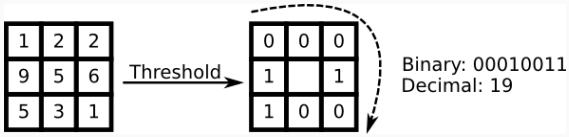

Fig. 18 LBP code creation example[1]

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

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 extract a histogram from each. Then, the spatially enhanced feature vector is obtained by concatenating the histograms, not merging them.

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

[1] Images extracted from OpenCV documentation.

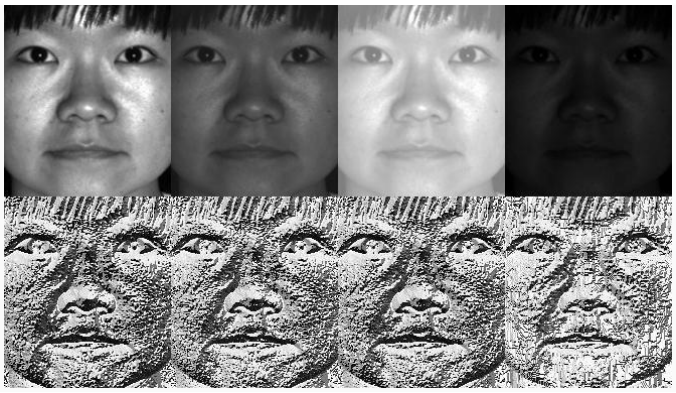

Fig. 19 LBP invariant to monotonic grayscale transformations

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 it provides 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:

$$score(model, x) = score\ (root, x) + \sum_{p \in parts} \max[score(p, y) - cost(p, x, y)] \quad (9)$$

The score at a particular position and scale within an image, $score(model, x)$, 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.

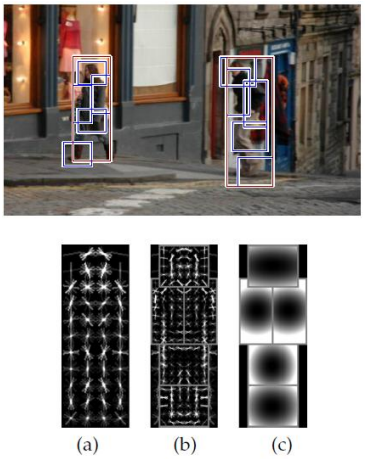

Fig. 20 Star model of a person category [41]

The training process use *latent SVM* (LSVM) to train models using partially labeled data, learn β, 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)

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

The loss function is convex in β 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_β is linear.

To detect objects in an image they compute an overall score for each root location according to the best possible placement of the parts. High-scoring root 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.

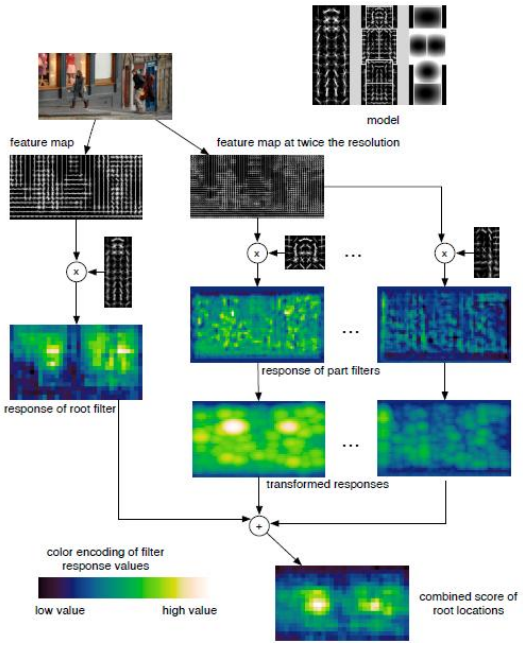

Fig. 21 Matching process for Deformable parts-based models approach [41]

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 lot 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 overlap of at least 50%.

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 representation of video content items to help users detect important objects in the video as well as providing 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 instance, 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 will use this marketing strategy to select the important content the user wants to see in our summary.

How do people actually choose what to watch? People tend to read a film overview for movies they want to watch. Another very interesting tendency is select those movies in which your favorite actor/actress appears. In TV watching behavior, for example, people consult program guides during viewing time to look for information.

A rapid grasp of the video content is a great source of information for every user. 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 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 Internet. For instance, if Brad Pitt appears in the movie, a user may want to see his filmography in order to know which other movie has Brad Pitt done. 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 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 requirement for fast and convenient content selection is 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 regions to achieve maximal efficiency. *Structural* requirements deal with the presentation of different regions within the result. Finally, *navigability* requirements concern to the source video browsing options.

The next subsections contain a complete list 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 naturally are interested in seeing the characters that are part of the video; therefore, frames including people should be preferred for being included in the final result.

Requirement P.2 *Fast understanding*
Although an image can contain storyline information, it has to contain frames with widely known relevant objects. It will allow users to rapidly grasp the content of the original video and they should be able to easily and quickly understand the included content.

Requirement P.3 *Visual variability*
Including different scenes within the video into the summary will allow our system to be more efficient. Furthermore, content and scene variability will help to maximize the whole source understanding of the abstraction.

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 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* (ROI) should not be distorted. This means that the chosen representative of each relevant object must be selected as the one with less distortion after processing. For instance, a large face will be less distorted than a small one.

3.2.4 Navigability requirements

Quickly browse the video is contained in navigability requirements in order to allow users selecting important scenes.

Requirement N.1 *Region boundaries*
Users must understand which the boundaries between different region representations are. This allows him to rapidly realize where he can browse different timestamp 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. The design is focused first to prioritize the requirements with the highest priority.

Time consuming computation is not always a priority because we are talking 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.

Requirement	Priority
P.1 People and main characters	1
P.2 Fast understanding	3
P.3 Visual variability	2
U.1 Non-repetition	1
U.2 Visual uniqueness	2
U.3 Characters uniqueness	2
S.1 Main characters excel	1
S.2 Style	2
N.1 Region boundaries	2
N.2 Metadata supplement	3

Table 1 Requirements overview and priorities

4

Solution approach

After having the requirements that our content-based video summary should fulfill in Chapter 3, we specify the solution approach in Chapter 4. In this chapter we further specialize and describe the implementation of the elements, constraints and functions that appeared in Chapters 2 and 3.

The rest of this chapter is structured as follows: Section 4.1 provides an overview of our solution approach 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

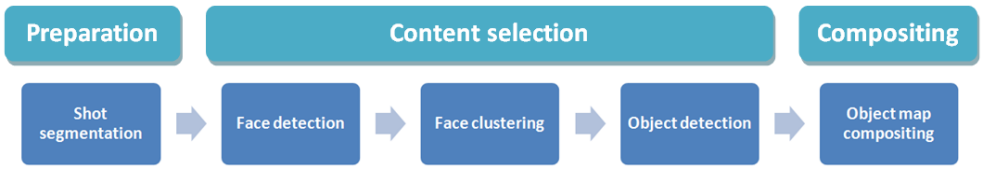

Fig. 22 Proposed system architecture

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

	Shot segmentation	Face detection	Face clustering	Object detection	Object map compositing
P.1 People and main characters		X	X		X
P.2 Fast understanding	X	X	X	X	X
P.3 Visual variability	X		X	X	X
U.1 Non-repetition	X				X
U.2 Visual uniqueness				X	
U.3 Characters uniqueness			X		
S.1 Main characters excel			X		
S.2 Style		X			X
N.1 Region boundaries					X
N.2 Metadata supplement				X	X

Table 2 Overview of requirements and architecture blocks

In the *preparation* step, the input video is sampled uniformly. Then, shot boundary detection selects frames 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 requirement that we want to solve at this stage. We cluster same person faces using face recognition algorithms. Then, we select largest faces in largest clusters as the representative to be included in the map. Secondly, for general object detection we perform a color-based similarity algorithm or maximum matching score results in order to group similar objects and select a representative for each detected object. In this step we aim at solving uniqueness requirements as well as structural ones.

Finally, in *composite* step, we create a visually attractive image composed by the most important content extracted in previous steps. We also want the object map to be as intuitive as possible to improve the browsing experience through the source video. The user must know what are the different regions and timestamps he

can navigate through. In this step we aim at fulfilling navigability requirements properly.

In next sections we describe all block's implementation and we mention third party software and approaches (described in detail in Chapter 2) we have used and those which have been rejected.

4.2 Shot segmentation

Video summary generation requires a temporal segmentation of the source video. This process is named *shot boundary detection* and *shot detection* by researchers and there exist significant amounts of methods. 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) of the video by plotting several shot representations (requirement U.1) trying to obtain as much information as possible in the summary.

4.2.1 Uniform sampling

Firstly, a uniform sampling extraction of the video frames is performed using *ffmpeg*[2] library wrapped by 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:

[2] http://www.ffmpeg.org/
[3] See Section 4.7 for further information about the development environment.
[4] http://opencv.org/

$$Sampling\ Rate = \min(\ \frac{N_0}{L_i} fps_i\ , fps_i\ )\ ,$$

(12)

where fps_i represents the acquisition frame rate in which the input video was generated, N_o the total number of frames we actually want to keep at the output to be processed by our shot detection algorithm, and L_i is the total number of frames of the input video. The second term corresponds is an upper bound and the first one is the downsampling rate ($\frac{N_0}{L_i}$) of the video sequence. We will vary N_o 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 Finding shot boundaries

The Software Studies Initiative[5] from the University of California in San Diego (UCSD) provides a very intuitive and simple source code from 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.

Binshtok and Greenshpan [13] 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 have found so far and it is described in detail in Section 2.2.2.

However, since Hausdorff and the neural network-based solutions require too much 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 face (see Section 4.3 Face detection) and object samples (see Section 4.5 Object detection) and generate more populated clusters (see Section 4.4 Face clustering). For these reasons, we decided to use initially 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.

[5] http://lab.softwarestudies.com/
[6] https://code.google.com/p/softwarestudies/

We have reduced the time redundancy of a video as the first milestone to be accomplished to finally create our summary. Now, we focus on the content selection stage. Particularly, we focus on the object which is the most relevant in a video, faces. In the next Section we will describe the development of our face detection algorithm.

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

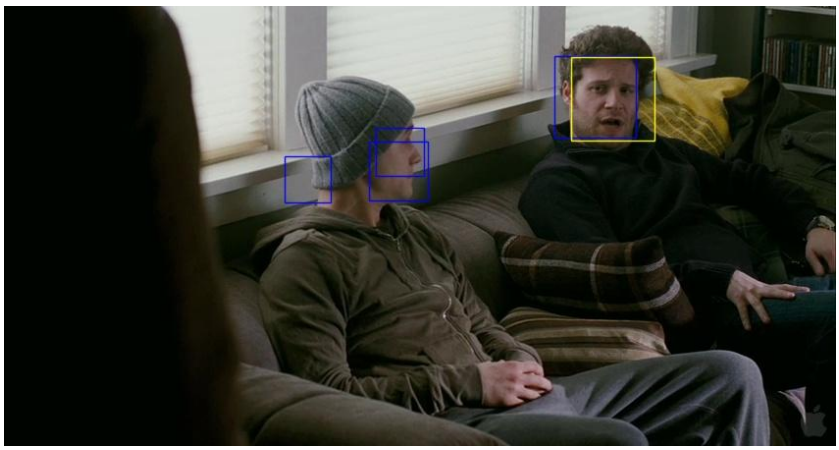

Fig. 23 Frontal (yellow) and profile (blue) detections over extracted keyframe

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 $V = \{f_1, \ldots, f_n\}$,
minimum detection size s_{min},
frontal cascade file F_{file},
profile cascade file P_{file};

```
1:  begin
2:      Initialize output ROI structures
3:      F = load classifier cascade(F_file)
4:      for i = 1, …, n do
5:          f_i ← get frame (f_i ϵ V)
6:          c⃗ ← detect faces (f_i, F)
7:          c⃗ ← remove faces (s_min)
8:          if (profile detection flag) then
9:              P = load classifier cascade(P_file)
10:             p1⃗ ← detect faces (f_i , P)
11:             p1⃗ ← remove faces (s_min)
12:             f_i' ← flip image (f_i)
13:             p2⃗ ← detect faces (fi', P)
14:             p2⃗ ← remove faces (s_min)
15:             overlap filtering (p1⃗, p2⃗)
16:             overlap filtering (c⃗, p1⃗)
17:             overlap filtering (c⃗, p2⃗)
18:         end
19:         release images (fi, fi')
20:     end for
21: end
```

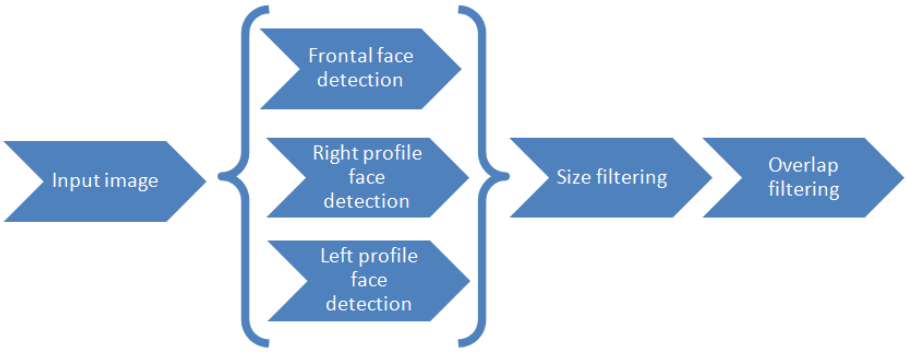

Fig. 24 Face detection stages architecture

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_frotalface_alt_tree.xml* and *haarcascade_profileface.xml.* The detection of frontal faces has proved to be reliable, with no feed 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

The output of this stage is a set of detected Regions of Interest (ROI) 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

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.

Fig. 27 Removed profile detection overlapping with frontal one

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 is not recommended and has to be retrained if someone wants to use it alone. 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.

Fig. 28 Face detection output

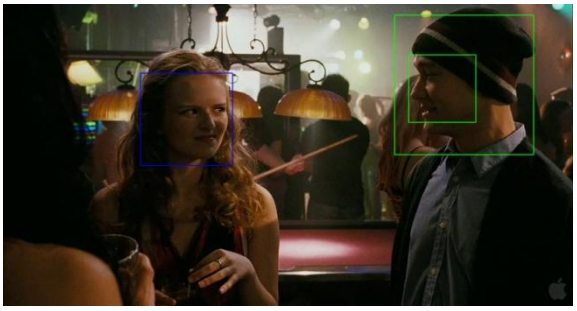

Fig. 29 Profile detection example

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).

4.4.1 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:

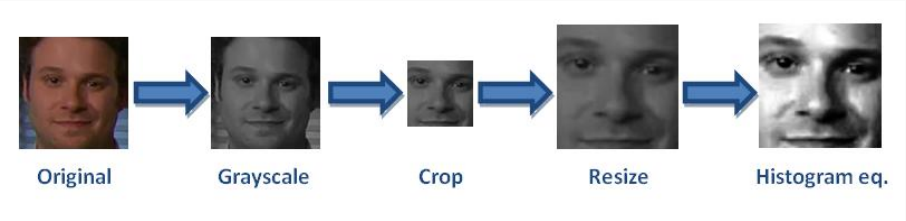

Fig. 30 Pre-processing facial images for face features extraction

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 deleted. 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…
6. …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.

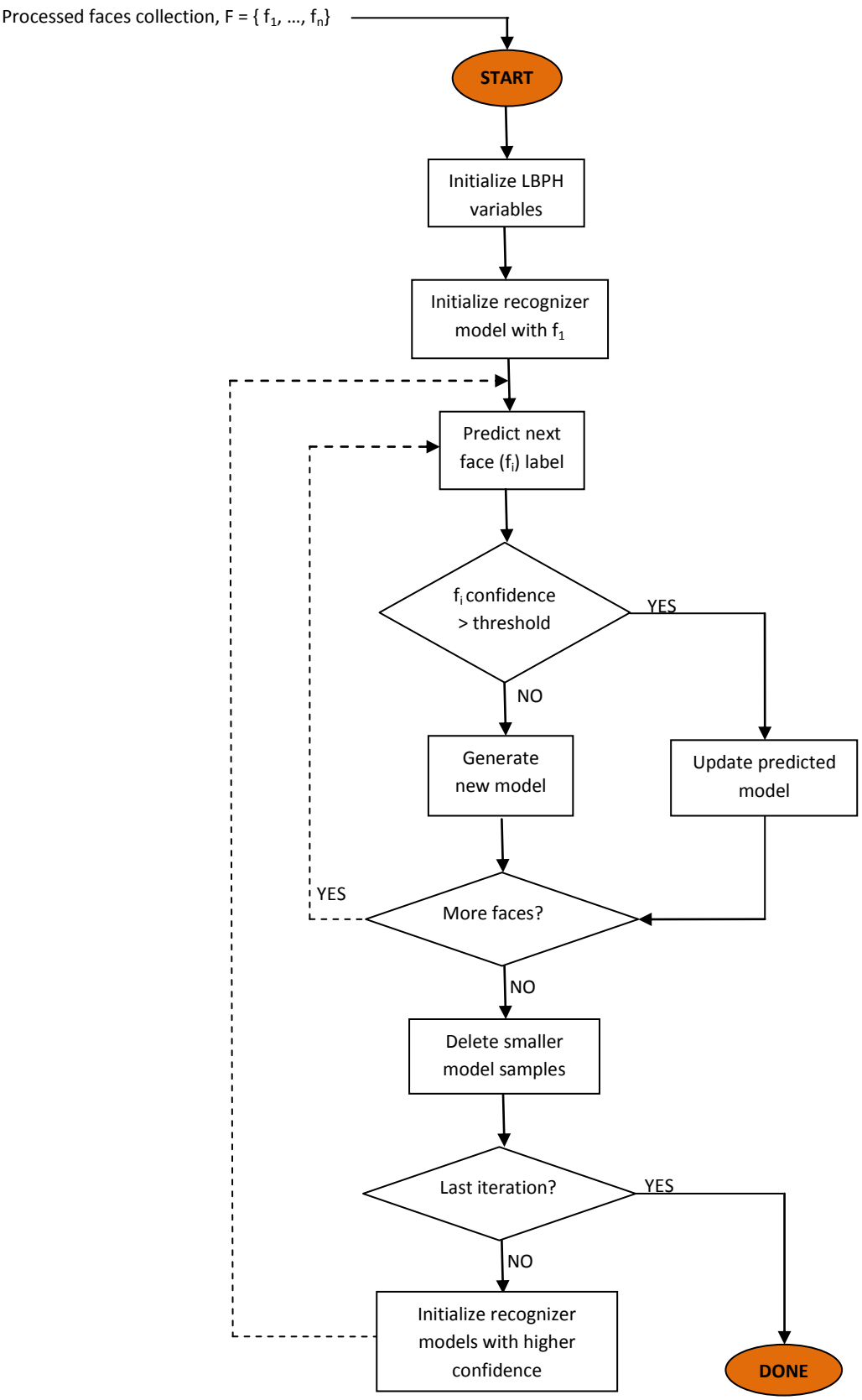

Fig. 31 Face clustering algorithm block diagram

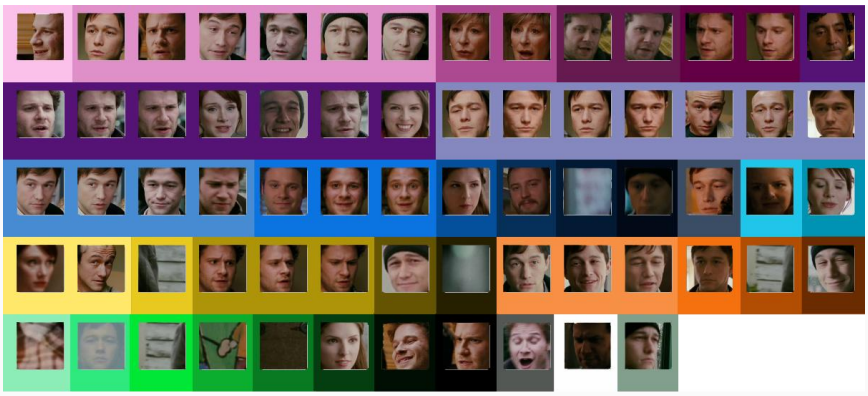

Fig. 32 First iteration of face clustering block

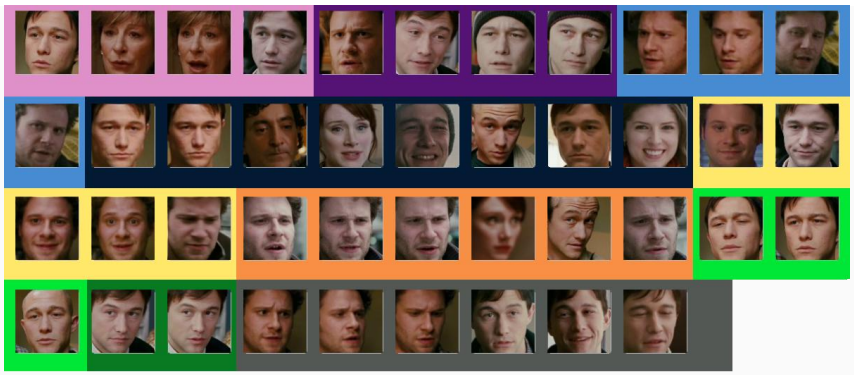

Fig. 33 Second iteration of face clustering block

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 but there are so many options that one could not provide a solution to all of them. The solution is to adopt a generic object detector and allow the final user to train it with the object of interest they prefer. The last content 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 content he wants. For this reason, multiple options have been developed to let the provider choose.

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 numerated 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 selected in *shot detection* block architecture.

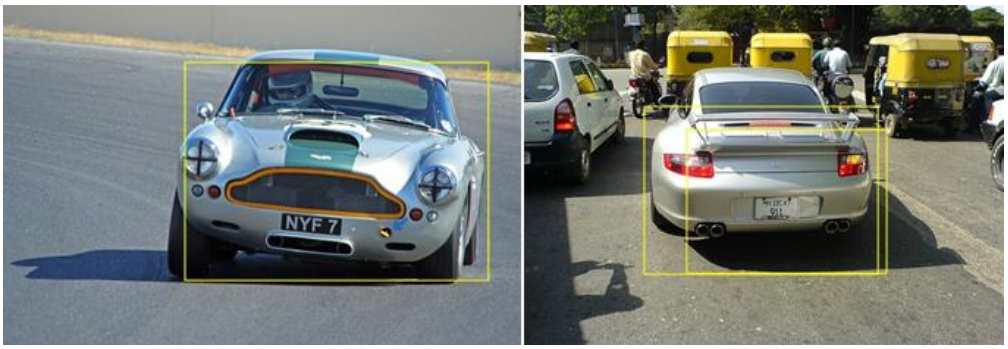

Fig. 35 Car object detection examples

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 the user training its own cascade classifiers, we present a simple 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 cascades we have developed an additional solution based on SURF features. 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.

Fig. 36 SURF training images

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.

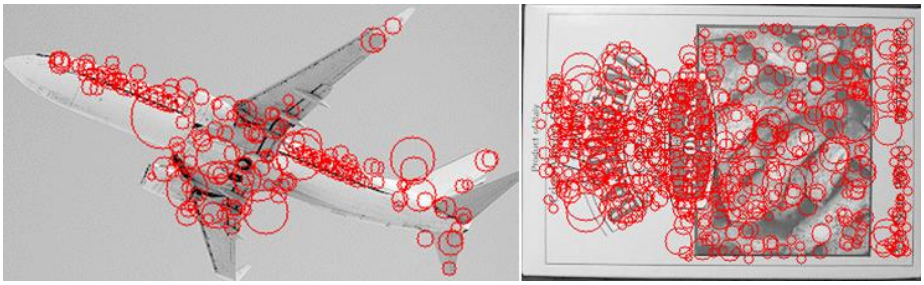

Fig. 37 SURF descriptors extraction

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 image with *cvExtractSURF*.
2. Find matching points between the training image's descriptors and the frame's descriptors using a laplacian filter for each one and finding its nearest neighbor using *Fast Approximate Nearest Neighbor* method. If the neighbor distance is lower than a preset threshold, the matching is positive.

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 the object 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 the SURF methods have.

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 ways 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.

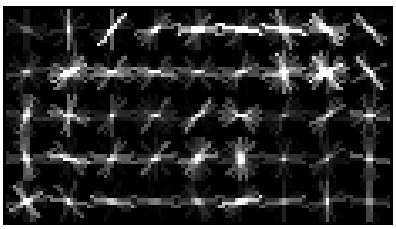

Fig. 39 Root filter of car model[7]

LibPaBOD[7] is built on top of two already used libraries: OpenCV[8] and MatIO[9]. Among other things, OpenCV library is used to handle images using the *IplImage* structure as well as *CvMat* and *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 executable. 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 false positives 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.

[7] http://www.uco.es/~in1majim/proyectos/libpabod/
[8] http://opencv.willowgarage.com/wiki/
[9] http://matio.sourceforge.net/

Fig. 40 Different view car detection using libpabod

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.

4.6.1 Content segmentation solution

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*.

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[10]. 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:

Fig. 41 Left: Source image. Right: Partition image

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 proposal is to create an oval model 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).

Fig. 42 Left: Detected faces with bounding boxes. Right: Positive oval marker in red, negative marker in blue

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,

[10] http://structuralsegm.sourceforge.net/

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

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

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 next figure.

Fig. 45 Face maps with largest face background representation

These presented results 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 segment the resulting 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.

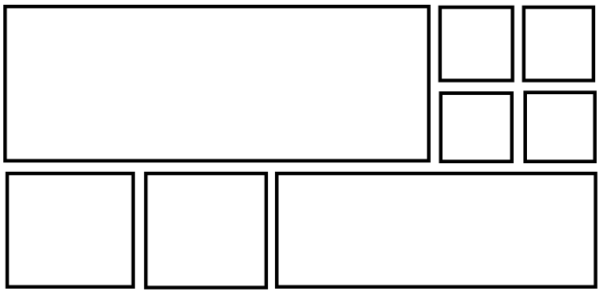

Fig. 46 Example of tile-based map with void regions

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 30 different 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:

Fig. 47 Tile-based composition examples

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 excel (requirement S.1) is accomplished by selecting different sizes to different regions depending on where the content comes and its relevance. Finally, the non-repetition requirement fulfillment (U.1) is possible labeling all added scenes to the supplement metadata 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 Development

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[11] 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[12] is an open source Software Configuration Management tool. This version control system has been used with the Subeclipse 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[13] 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[14] 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[15]. 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 utilities to train Haar cascade classifiers:

- *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.

[11] http://www.eclipse.org
[12] http://subversion.apache.org/
[13] http://java.com/
[14] http://opencv.org/
[15] http://www.willowgarage.com/

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[16]. 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.
- 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[17] 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

Graphical Annotation Tool (GAT)[18] provides an interface to create ontologies and to label positive, negative and neutral instances for each ontology class [47]. 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.

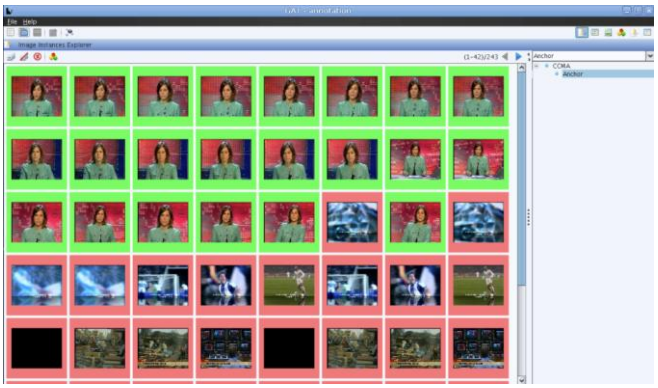

Fig. 48 GAT user interface

[16] https://code.google.com/p/javacv/
[17] http://www.ffmpeg.org/
[18] http://upseek.upc.edu/gat/

4.7.1 Using the application

The Java software released is easy to use. The user does not need to install OpenCV and JavaCV libraries because they are also released with the code. All compiled libraries are saved into the "*user.home"* directory if they do not exist. 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:

```
>> 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 options 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, 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.

5

Evaluation

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 the *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[19], 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:

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 **H0** into four more specific ones:

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.
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.
H3. Our approach better represents main characters than the *uniform sampling* method but worse than the *manual selection* method.
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.

5.2 Method

Different methods to generate video summaries are compared (object map, uniform sampling, and manual selection). Each subject 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 disadvantage is

[19] http://www.imdb.com/

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 36 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[20] 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, *Fast & Furious* summary was created using a car model and *Django Unchained* video was processed using a horse model running the included parts-based object detector (see Chapter 4, Section 4.5.3).

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

Table 3 Video items used in the user study

[20] http://trailers.apple.com/

5.5 Procedure

The participants were given a web-based survey[21] 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 and 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 should satisfy 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**.

[21] 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).

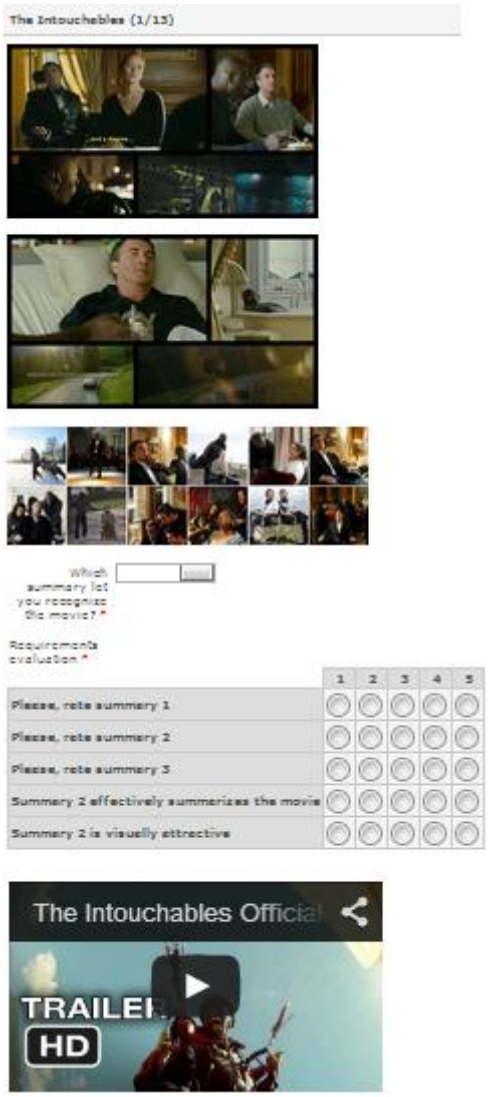

Fig. 49 Web-based evaluation survey

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.

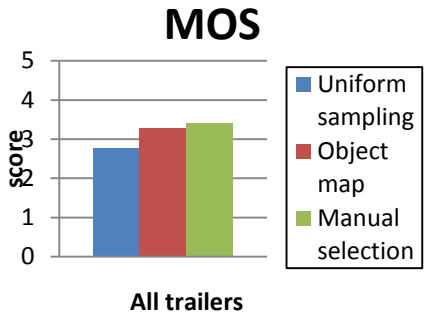

Fig. 50 Global rating of the summaries

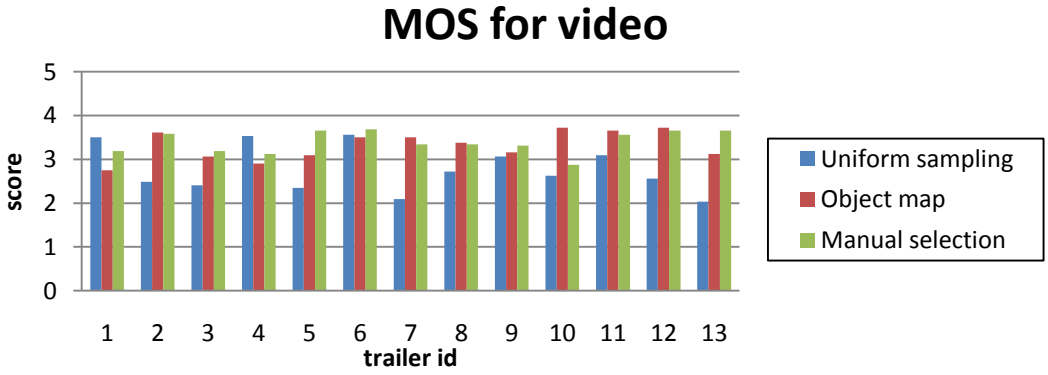

Fig. 51 Individual rating of the summaries

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 high 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.

Fig. 52 Object map summary for trailer 1, The Intouchables

Fig. 53 Uniform sampling summary for trailer 1, The Intouchables

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.

Fig. 54 Object map summary for trailer 7, The Fast and the Furious

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, **Q1**. Fig. 55 shows the average recognition score for each technique with the following interpretation: uniform sampling (a), uniform sampling + object map (b), and uniform sampling + object map + manual selection (c).

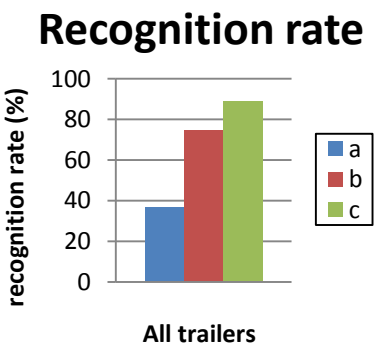

Fig. 55 Global recognition rate

In **¡Error! No se encuentra el origen de la referencia.** 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 user could recognize the movie. The 100% is not achieved with (c) because some users did not recognize some movies.

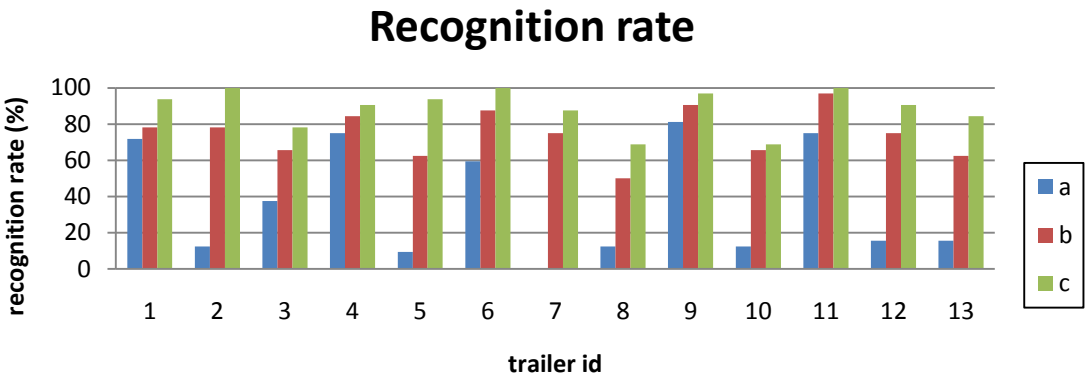

Fig. 56 Individual recognition rate

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.

Fig. 57 Uniformly sampled summary of trailer 4, Dark Shadows

Fig. 58 Uniformly sampled summary of trailer 9, Resident Evil 5: Retribution

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 Figure 11 while individual results are shown in Fig. 60.

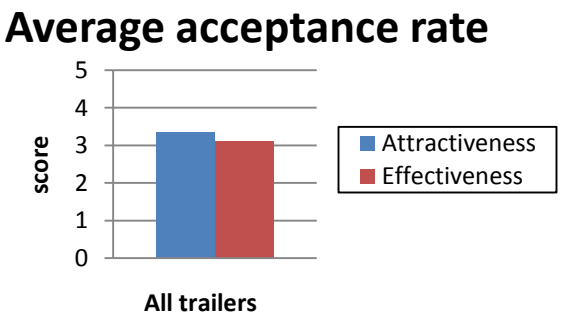

Fig. 59 Global acceptance rate

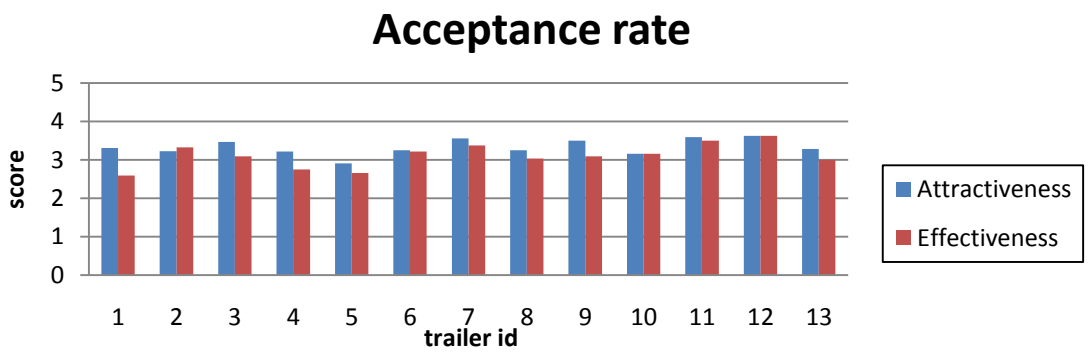

Fig. 60 Individual acceptance rate

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.

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

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. Even 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 state by Technische Universität Wien (TUWien) and Universitat Politècnica de Catalunya (UPC). We aim at providing solution to 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 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 rues 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 sourforge site[22].

[22] http://sourceforge.net/p/objectmaps

Future work

At this point, we identify four 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.
- 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 adaptative 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 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.

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

[1] Dufaux, F. Key frame selection to represent a video. *Proceedings 2000 International Conference on Image Processing.* Cat No00CH37101 275–278 vol.2 (2000). IEEE. doi:10.1109/ICIP.2000.899354

[2] Ma, Y.-F., Lu, L., Zhang, H.-J., & Li, M. (2002). A user attention model for video summarization. *Proceedings of the tenth ACM international conference on Multimedia - MULTIMEDIA '02*, 533. doi:10.1145/641113.641116

[3] Pfeiffer, S., Lienhart, R., Fischer, S., & Effelsberg, W. (1996). Abstracting Digital Movies Automatically. *Journal of Visual Communication and Image Representation*, *7*(4), 345–353. doi:10.1006/jvci.1996.0030

[4] Ponceleon, D., Amir, A., Srinivasan, S., Syeda-Mahmood, T., & Petkovic, D. (1999). CueVideo: Automated Multimedia Indexing and Retrieval. *ACM Multimedia 1999* (p. 199). ACM Press.

[5] Smith, M. A., & Kanade, T. Video skimming and characterization through the combination of image and language understanding techniques. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 775–781, 1997.

[6] Christel, M. G., Smith, M. A., & Taylor, C. R. Evolving video skims into useful multimedia abstractions. In *Proceedings of Conference on Human Factors in Computing Systems*, pages 171–178, 1998.

[7] Hauptmann, A. G., & Smith, M. A. Text, speech, and vision for video segmentation: The informedia project. In Proceedings *of the AAAI Fall Symposium on Computer Models for Integrating Language and Vision*, pages 213–220, 1995.

[8] Ngo, C.-W., Ma, Y.-F., & Zhang, H.-J. Video summarization and scene detection by graph modeling. , *15 IEEE Transactions on Circuits and Systems for Video Technology* 296–305 (2005). IEEE. doi:10.1109/TCSVT.2004.841694

[9] Gao, Y., & Dai, Q.-H. (2008). Shot-based similarity measure for content-based video summarization. *15th IEEE International Conference on Image Processing* 2512–2515 (2008). IEEE. doi:10.1109/ICIP.2008.4712304

[10] Frey, B. J., & Dueck, D. (2007). Clustering by passing messages between data points. *Science*, *315*(5814), 972–976.

[11] Zhao, W., Wang, J., Bhat, D., Sakiewicz, K., Nandhakumar, N., & Chang, W. (1999) Improving color based video shot detection. 2 *Proceedings IEEE International Conference on Multimedia Computing and Systems* 752–756 (1999). IEEE Comput. Soc. doi:10.1109/MMCS.1999.778579

[12] Kasturi, R., Strayer, S. H., Gargi, U., & Antani, S. (1996). An Evaluation of Color Histogram Based Methods in Video Indexing.

[13] Binshtok, M., & Greenshpan, O. (2006). Segmentation of Video Incorporating Supervised Learning.

[14] Rasheed, Z., & Shah, M. (2005). Detection and representation of scenes in videos. 7 *IEEE Transactions on Multimedia* 299–1105 (2005). IEEE. doi:10.1109/TMM.2005.858392

[15] Taniguchi, Y., Akutsu, A., Tonomura, Y., & Hamada, H. (1995). An intuitive and efficient access interface to real-time incoming video based on automatic indexing. *Proc of ACM Multimedia95* (pp. 25–33). ACM SIGMM, SIGCHI, SIGGRAPH ACM Press.

[16] Ferman, A. M., & Tekalp, A. M. (1998). Efficient Filtering and Clustering Methods for Temporal Video Segmentation and Visual Summarization. *Journal of Visual Communication and Image Representation*, *9*(4), 336–351. doi:10.1006/jvci.1998.0402

[17] Zhang, H. J., Wu, J., Zhong, D. I., & Smoliar, S. W. (1997). An integrated system for content-based video retrieval and browsing. *Pattern Recognition*, *30*(4), 643–658. doi:10.1016/S0031-3203(96)00109-4

[18] Ngo, C.-W., Pong, T.-C., & Chin, R. T. (2001) Video partitioning by temporal slice coherency. , 11 *IEEE Transactions on Circuits and Systems for Video Technology* (2001). doi:10.1109/76.937435

[19] Cernekova, Z., Pitas, I., & Nikou, C. (2006) Information theory-based shot cut/fade detection and video summarization. , 16 *IEEE Transactions on Circuits and Systems for Video Technology* 82–91 (2006). IEEE. doi:10.1109/TCSVT.2005.856896

[20] Yeung, M., Yeo, B.-L., & Liu, B. (1996). Extracting story units from long programs for video browsing and navigation. *Multimedia Computing and*

Systems 1996 Proceedings of the Third IEEE International Conference on (Vol. 1996, pp. 296–305). doi:10.1109/MMCS.1996.534991

[21] Zhuang, Y., Rui, Y., Huang, T. S., & Mehrotra, S. (1998). Adaptive key frame extraction using unsupervised clustering. 1 *Proceedings 1998 International Conference on Image Processing* ICIP98 Cat No98CB36269 866–870 (1998). IEEE Comput. Soc. doi:10.1109/ICIP.1998.723655

[22] Rui, Y., Huang, T. S., & Mehrotra, S. (1999). Constructing table-of-content for videos. *Multimedia Systems*, *368*(5), 359–368. doi:10.1007/s005300050138

[23] Aner, A., & Kender, J. R. (2002). Video Summaries through Mosaic-Based Shot and Scene Clustering. *Proceedings of the 7th European Conference on Computer Vision Copenhagen Denmark*, *LNCS 2353*, 388–402.

[24] Tobita, H. (2010). DigestManga : Interactive Movie Summarizing through Comic Visualization. *Movie*, 3751–3756. doi:10.1145/1753846.1754050

[25] Lim, Y., Uh, Y., & Byun, H. (2011) Plot preservation approach for video summarization. 2011 *IEEE International Conference on Systems Man and Cybernetics* 67–71 (2011). IEEE. doi:10.1109/ICSMC.2011.6083644

[26] Nguyen, C., Niu, Y., & Liu, F. (2012). Video Summagator : An Interface for Video Summarization and Navigation. *Proceedings of the 2012 ACM annual*, 647–650. doi:10.1145/2207676.2207767

[27] Chen, T., Lu, A., & Hu, S. (2011). Visual Storylines : Semantic Visualization of Movie Sequence. *Computers & Graphics*, Volume 36, Issue 4, June 2012, Pages 241-249, ISSN 0097-8493

[28] Chiu, P., Girgensohn, A., & Liu, Q. L. Q. (2004). Stained-glass visualization for highly condensed video summaries. , 3 *2004 IEEE International Conference on Multimedia and Expo ICME IEEE* Cat No04TH8763 2059–2062 (2004). IEEE. doi:10.1109/ICME.2004.1394670

[29] Wang, T., Mei, T., Hua, XS., Liu, XL., & Zhou, HQ. (2007). Video collage: A novel presentation of video sequence. *IEEE International Conference on Multimedia and Expo 2007;:1479–82*. doi:10.1109/ICME.2007.4284941.

[30] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR 2001*, *1*(C), I–511–I–518. doi:10.1109/CVPR.2001.990517

[31] Bay, H., Tuytelaars, T., & Van Gool, L. (2006). SURF: Speeded Up Robust Features. (A. Leonardis, H. Bischof, & A. Pinz, Eds.)*Computer Vision–ECCV 2006*, *3951*(3), 404–417.

[32] Belhumeur, P. N., Hespanha, J., & Kriegman, D. (1997). Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19, 7 (1997), 711–720.

[33] Meena, K., & Suruliandi, A. (2011). Local binary patterns and its variants for face recognition. 2011 *International Conference on Recent Trends in Information Technology ICRTIT 782–786* (2011). IEEE. doi:10.1109/ICRTIT.2011.5972286

[34] Jain, V., & Learned-miller, E. (2010). FDDB : A Benchmark for Face Detection in Unconstrained Settings. *Technical Report UM-CS-2010-009, Univ. of Massachusetts, Amherst*.

[35] Li, S., Zhu, L., Zhang, Z., Blake, A., Zhang, H., & Shum, H. (2002). Statistical learning of multi-view face detection. (A. Heyden, G. Sparr, M. Nielsen, & P. Johansen, Eds.) *Proc European Conf on Computer Vision*, *2353*, 67–81. doi:10.1007/3-540-47979-1_5

[36] Wu, B., Ai, H., Huang, C., & Lao, S. (2004) Fast rotation invariant multi-view face detection based on real Adaboost. , *Sixth IEEE International Conference on Automatic Face and Gesture Recognition 2004 Proceedings* 79–84 (2004). IEEE. doi:10.1109/AFGR.2004.1301512

[37] Wu, B., & Nevatia, R. (2007). Cluster Boosted Tree Classifier for Multi-View, Multi-Pose Object Detection. , *IEEE 11th International Conference on Computer Vision* (2007) 1–8 (2007). Ieee. doi:10.1109/ICCV.2007.4409006

[38] Zhang, C., & Zhang, Z. (2009). Winner-Take-All Multiple Category Boosting for Multi-view Face Detection.

[39] Felzenszwalb, P., McAllester, D., & Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. *IEEE Conference on Computer Vision and Pattern Recognition (2008)*, *08*([9), 1–8. doi:10.1109/CVPR.2008.4587597

[40] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(9), 1627–1645.

[41] Felzenszwalb, P. F., Girshick, R. B., & McAllester, D. (2010). Cascade object detection with deformable part models. 460 *Computer Vision and Pattern Recognition CVPR 2010 IEEE Conference on 2241–2248* (2010). IEEE. doi:10.1109/CVPR.2010.5539906

[42] Lienhart, R., & Maydt, J. (2002). An extended set of Haar-like features for rapid object detection. *Proceedings International Conference on Image Processing*, *1*(2002), I–900–I–903. doi:10.1109/ICIP.2002.1038171

[43] Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, *60*(2), 91–110. doi:10.1023/B:VISI.0000029664.99615.94

[44] Consularo, L. A., Cesar, R. M., & Bloch, I. (2007). Structural Image Segmentation with Interactive Model Generation. 6 *2007 IEEE International Conference on Image Processing* 45–48 (2007). IEEE. doi:10.1109/ICIP.2007.4379517

[45] Noma, A., Pardo, A., & Cesar, R. M. (2010). Structural matching of 2D electrophoresis gels using deformed graphs. *Pattern Recognition Letters*, *32*(1), 3–11. doi:10.1016/j.patrec.2010.02.016

[46] Noma, A., Graciano, A. B. V., Cesar Jr, R. M., Consularo, L. A., & Bloch, I. (2012). Interactive image segmentation by matching attributed relational graphs. *Pattern Recognition*, *45*(3), 1159–1179. doi:10.1016/j.patcog.2011.08.017

[47] Giro-i-Nieto, X., Camps, N., & Marques, F. (2009). GAT, a Graphical Annotation Tool for semantic regions. *Multimedia Tools and Applications 2009*.

[48] Over, P., Smeaton, A. F., & Kelly, P. (2007). The trecvid 2007 BBC rushes summarization evaluation pilot. *Proceedings of the international workshop on TRECVID video summarization TVS 07*, 1–15. doi:10.1145/1290031.1290032

[49] Over, P., Smeaton, A. F., & Awad, G. (2008). The trecvid 2008 BBC rushes summarization evaluation. *Proceeding of the 2nd ACM workshop on Video summarization TVS 08* (pp. 1–20). ACM Press. doi:10.1145/1463563.1463564

[50] Zhang, L., Kalashnikov, D. V, & Mehrotra, S. (2013). A Unified Framework for Context Assisted Face Clustering. *ICMR ´13*, 9–16.

A

Test data

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[23]

[23] http://www.youtube.com/watch?v=9B1bXeNUWGc

Fig. 62 50/50[24]

Fig. 63 The Twilight saga - Breaking Dawn part 1[25]

Fig. 64 Dark Shadows[26]

[24] http://www.youtube.com/watch?v=mMaJET7mD0M
[25] http://www.youtube.com/watch?v=PQNLfo-SOR4
[26] http://www.youtube.com/watch?v=wpWvkFlyl4M

Fig. 65 The dictator[27]

Fig. 66 Django Unchained[28]

Fig. 67 The fast and the furious[29]

[27] http://www.youtube.com/watch?v=DS2lURW4JSI
[28] http://www.youtube.com/watch?v=eUdM9vrCbow
[29] http://www.youtube.com/watch?v=2TAOizOnNPo

Fig. 68 The Intouchables[30]

Fig. 69 The Lord of the Ring - The Fellowship of the Ring[31]

Fig. 70 The Matrix[32]

[30] http://www.youtube.com/watch?v=34WIbmXkewU
[31] http://www.youtube.com/watch?v=Pki6jbSbXIY
[32] http://www.youtube.com/watch?v=r1GrMAqwWcI

Fig. 71 Mirror, Mirror[33]

Fig. 72 Resident Evil 5: Retribution[34]

Fig. 73 Star Wars: Episode I - The Phantom Menace[35]

[33] http://www.youtube.com/watch?v=YgbH05rQx1s
[34] http://www.youtube.com/watch?v=HYuxE3YetQo
[35] http://www.youtube.com/watch?v=1dWA9DwDQpM

B

Training Object detectors with OpenCV and Pascal VOC

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.

Fig. 74 20 VOC object classes examples

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[36].

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.PascalTrainingData-Generator.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_triancascade` tools.

[36] 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. `PascalTrainingData-Generator.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.

```
positive/img1.jpg 1 140 100 45 45
positive/img2.jpg 2 100 200 50 50 30 25 25
positive/img3.jpg 1 0 0 20 20
```

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 `openc_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 min_hit_rate^number_stages).
- *-maxFalseAlarmRate:* Maxima desired false alarm rate for each stage of the classifier. Overall false alarm rate may be estimated as (max_false_alarm_rate^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.