(Semi) Automatic Infinite Entertainment System

Diogo Serrano (45017) *
Departamento de Informática
FCT/UNL
Caparica, Portugal
dm.serrano@campus.fct.unl.pt

Diogo Silvério (45679)
Departamento de Informática
FCT/UNL
Caparica, Portugal
d.silverio@campus.fct.unl.pt

***Abstract*—With the great advancements in the field of augmented reality there has been an increase in popularity with entertainment systems based on this technology. These systems require not only powerful algorithms to detect the real world environment and translate it into data perceivable by a computer but also powerful video processing algorithms that allow the system to mimic the way a human being perceives the real world. In this paper we present a system that builds an infinite entertainment system based on the environment captured by a camera.**

***Index Terms*—video entertainment, video features, video interface, face detection**

I. Introduction

In this report we present the initial specification of the system and a study of the related work. The goal of this project is to create a system capable of providing continuous video content based on the environment in which a user is inserted. The application has 2 available modes: one for the operator and another for the user. The operator mode allows an operator to explore a collection of videos, check their correspondent metadata and control the content provided to the user by creating playlists of videos to be displayed based on the data retrieved from the user's environment. This data is captured on the user's mode of the application through a camera and is then processed based on, for example, the amount of people in front of the camera and displays videos that will change based on the user's current environment.

II. Related Work

In order to build a high quality system we explored some of the solutions provided to some of the problems in the areas of video feature extraction, similarity between videos, video interfaces and video content exploration by others.

A. *Video Exploration: From Multimedia Content Analysis to Interactive Visualization [1]*

As mentioned in [1], "Video analysis is a time consuming process because it implies time linear reading". In order to solve this problem the authors suggest 3 solutions to this problem based on the volume of video resources, type of granularity of the content, and the originality of the related visualizations.

The *Stream Explorer* allows a user to explore one video with the shortest segment being one single frame and the interface uses a tape recorder metaphor to allow the user to understand that they have in fact access to this level of granularity. This kind of interface could be used on our system to allow the operator to cut videos in a future version.

The *TV Programme Explorer* displays a collection of videos similar to the one that is currently being watched under the assumption that "[...] visual redundancies, which are the result of some editing process, reflect most of the time higher levels of semantic" [1]. This is quite an interesting approach to take into consideration seeing as the video content being displayed is based on its characteristics such as topic, colour, texture, rhythm or keywords.

The *CollectionExplorer* provides an overview of the size and richness of the available video content gathering documents semantically similar. Applying this feature to our system would allow the operator to easily choose videos with similar semantic content without having to inspect the metadata of a video that normally does not hold any semantic value.

B. *VFerret: Content-Based Similarity Search Tool for Continuous Archived Video [2]*

VFerret is a content-based similarity search tool for continuous archived video that does not depend on attributes or annotations to search desired data from long-time archived video and allows users to perform content-based similarity search using visual and audio features, and also combine content-based similarity search with traditional search methods [2]. According to the authors the tool provides an average precision of 0.79 when combining both visual and audio features [2], which makes this tool an interesting addiction to the system for when an operator is building a playlist since it would ease the video searching process allowing a semantic based search of video content. It also fits very nicely with the system because the videos metadata is already being extracted and stored.

C. *Advanced User Interfaces for Dynamic Video Browsing [3]*

In this paper we're presented with 3 different user interfaces for dynamic video browsing, each trying to address a problem with the common video browsing interfaces. The *ZoomSlider* allows the user to zoom in and out of the slider's scale while

*Did not participate in the 2nd phase of the project.

browsing the video in order to control the scrolling granularity. This seems like a good approach since it's very simple to use and allows the user to control the level of granularity they wish to use. The *NLslider* uses a non linear slider scale in order to solve scaling problems allowing the entire scale to be visible within the application window. However, the non linear slider scale might be confusing to users who are not used to it and for that reason we don't plan on using this approach on our system. The *Elastic Skimming* interface is based on the concept of elastic interfaces and enables the user to control the scrolling speed without modifying the scale of the slider. This seems like an interesting approach for devices that support touchscreen technology. However, since our system is being developed for computers, which mostly do not support this technology, we won't be applying it to our project.

III. Specification

The goal of our system is to continuously provide video content based on some gathered context information such as if someone is watching the screen or how many people are paying attention to it. One of the specifications of the system is that the video content should be selected based on its characteristics such as topic, colour, texture, rhythm or keywords.

As said in the system requirements the system will be separated in two main parts: one for sensing the environment, count and detect people and another to extract characteristics of the video streams and use them to select what is playing. The system will also include transitions between videos.

There will be two main modes, the first one being the display of the resulting video in a public screen and the second being the setup and configuration mode for the system. The first mode will only display the selected video whereas the second mode allows the operator to see the available video library, providing an assorted amount of options related to the videos metadata and playlist configurations.

A. *User Mode Interface*

This Interface corresponds to the first of the two main modes mentioned previously, the Video Player Interface which plays the sequence of several video clips from a selected playlist (Fig.1). Play/stop/pause/resume controls will also be available options for the user in charge of the screen (using the keyboard).

B. *Operator Mode Interface*

When the application is opened by the operator, it starts in the Thumbnails Page, where the user is presented with all the available playlists to select which ones they wish to present to the public screen. Each video presented in this interface is going to be represented by a moving icon (a video thumbnail). The Operator is also presented with the option to create a new playlist, at the top right corner of the screen, and with the option of editing the metadata of a video specified by the operator (Fig.2).

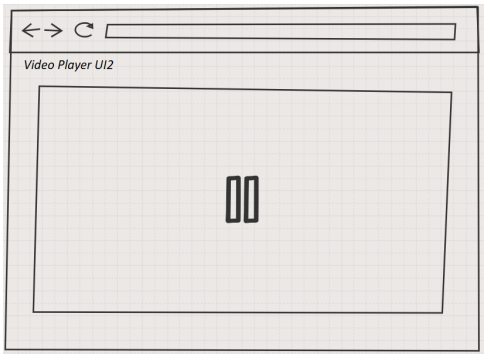

Fig. 1. Video Player Interface UI2

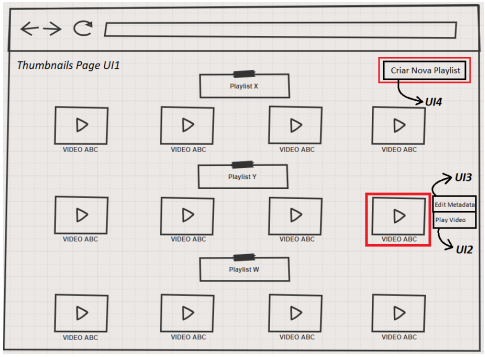

Fig. 2. Playlist Thumbnails Page Interface UI1

The Video Library is also present in this same page and can be seen by scrolling down on the window. When no playlists have been created the Video Library can be seen without the need of scrolling. Below we present an early video library prototype.

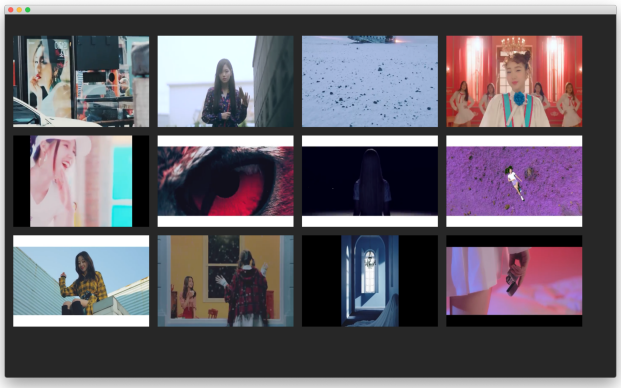

Fig. 3. Video Library prototype

In the previously shown Playlist Thumbnails Interface (Fig. 2), if the Operator chooses to create a new playlist, they will be redirected to the Creation Form Interface. Here the Operator has to choose the name of the new playlist and then select 'Create List' (Fig. 4). After this the Operator will be redirected back to the Playlist Thumbnails Page Interface and

Video Library (Fig. 2).

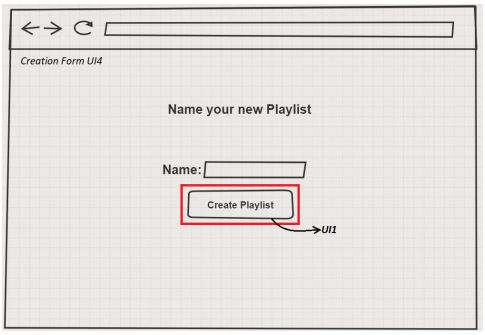

Fig. 4. (Playlists) Creation Form Interface UI4

On the other hand, if the Operator chooses to edit a video metadata, they will be redirected to the Video Metadata Editor Interface, where the Operator is presented with the possibility of editing the videos metadata, e.g. keywords, and also be able to change the playlist in which the video is included. Whenever the Operator is satisfied with all the changes they've made, the Operator should select the button 'End Changes' (Fig. 5) in order to save them. They will then be redirected back to the Playlist Thumbnails Page Interface and Video Library (Fig. 2).

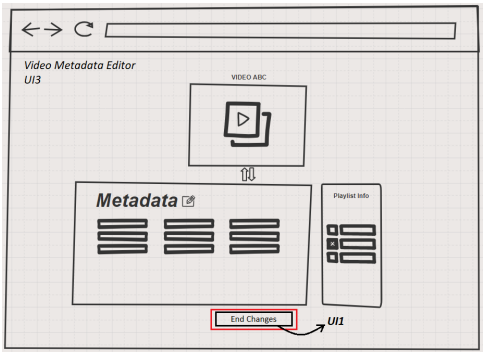

Fig. 5. Video Metadata Editor Interface UI3

To end the system specification we present a simple storyboard presenting the different interfaces provided by the system and how a user can travel between them.

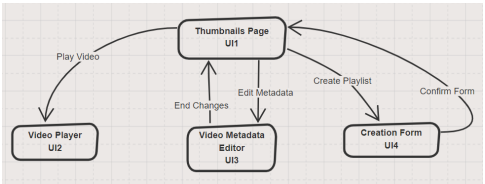

Fig. 6. Interface storyboard

IV. Algorithms and techniques (description of the processing algorithms and visualization techniques)

Several techniques were used in order to perform video processing, manipulate image and pixel files to obtain metadata. The most basic technique would be to obtain the dominant colour of a video which is done by iterating every pixel of every frame of the video and the value of each channel of the three channels (red, green, blue) of each of pixel. Then once we're done iterating the video the mean of is calculated. To calculate the luminance of a video the YUV colour model is used to separate the luminance from the crominance7. In order to convert the RGB values given by the video into luminance, the following equation is used: $luminance = 0.2125 * red + 0.7154 * green + 0.0721 * blue$.

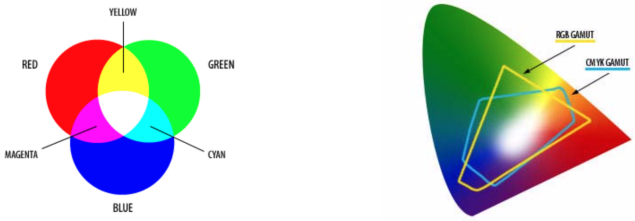

Fig. 7. Difference between the RGB and the YUV colour models [4].

To obtain a video's edge distribution we iterate over every pixel of every frame and begin by converting the pixels to grey scale as colour doesn't provide any information about edges. Then we apply five filters to the frame separately that perform a convolution. Each of these filters have different kernels to represent different directions of the edges or even no direction but all compute edges as the sum of the values of the kernels always equal to zero. After the filters are applied we obtain images where high values represent edges and create histograms by counting the values of the image. To improve results and trim false positives edges we apply a threshold and only count pixels with a value above 200.

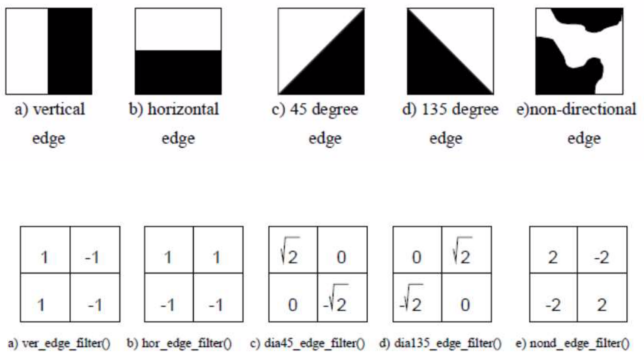

Fig. 8. Different edge kernels and their respective meaning [4]

The texture characteristics of a video were only calculated for the first frame to save time and intense computing when retrieving video metadata. The algorithm is exactly the same as the previous one used for the edges except for the kernel used in the convolution computed by the filter. Now we apply the Gabor kernel instead which is characterised by the following equation:

$$G(x,y) = e^{-\frac{x'^2+\Upsilon^2 y'^2}{2\sigma^2}} cos(2\pi\frac{x'}{\lambda} + \psi),$$
$$x' = x\cos\theta + y\sin\theta,$$
$$y' = -x\sin\theta + y\cos\theta$$

The filter was used 24 different times with 24 different combinations of arguments to obtain 24 different textures.

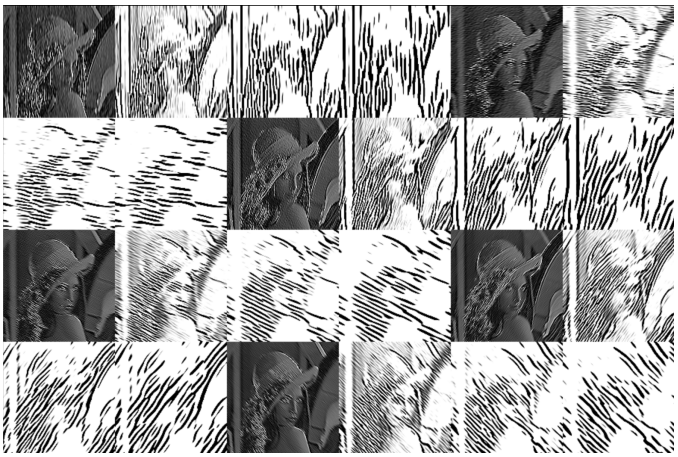

Fig. 9. Result of applying the 24 different Gabor filters in an image.

The video's rhythm is calculated with the help of the edge distributions. For each frame the edge distribution is calculated and then a histogram subtraction is performed for every resulting image. The difference between the 2 edge images will indicate either low movement or rhythm if the result is low or high movement or rhythm if the result is high.

To compute the number of faces found in a video, just like before, filters are applied. In this case Haar features are used and work very similarly to the edge kernels but now all the features are applied to a small zone of the image before moving to the next, sweeping the entire image. Every time the features aren't moving out of a zone of the image is because they might be succeeding at extracting a human face. Once a certain numbers of them fail the probability of existing a face inside that zone is reduced and they soon move on to the next one.

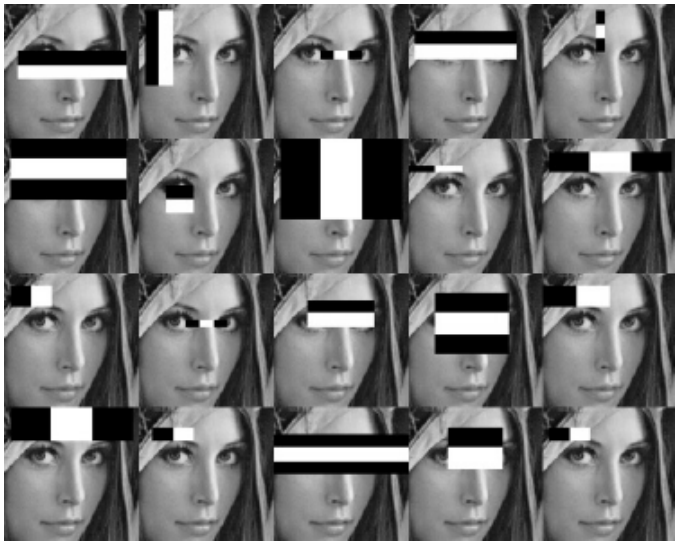

Fig. 10. Different stages of Haar features being applied to an image [5].

Last we calculate the number of times an image appears in a video frame. This process is divided into three main parts: keypoint detection, descriptor extraction and descriptor matching. The keypoint detection finds the vertices of an image that might be important to represent the object that we're trying to find. The descriptor extraction extracts data from the image that characterises it, namely the image's gradient. In the end the descriptor matching tries to match both the descriptor of the video frame and the image of the object we're trying to locate by using distance metrics.

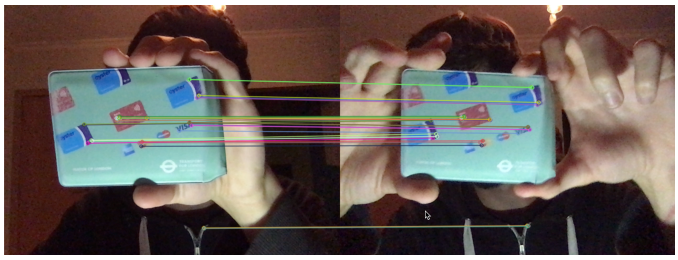

Fig. 11. Object detection being performed.

V. Application Implementation

Unfortunately, due to the lack of a project partner for the 2nd phase, there was no time for the application code to be organised into different files and so it is all implemented into a single one. Nevertheless all features, algorithms and techniques were implemented compensating for such flaw. The application was implemented in c++ using the openFrameworks

toolkit. Besides the core, the included addons ofxXmlSettings and ofxOpenCv and the extra addons ofxCv and ofxDatGui were also used. The ofxXmlSettings simplifies the process of writing and reading XML files that were used to save video metadata and video playlists information. The ofxOpenCv integrates the computer vision library with openFrameworks. The ofxCv is an extra addon that simplifies the interaction between the two previous addons by providing several util like functions and the ofxDatGui is an extra addon based on a well known javascript graphical user interface that was used to improve and facilitate the development of the application's GUI. The User and Operator mode is selected on the welcome page of the application. The User mode is visually simple as it only displays a single video but it's constantly sensing its environment by detecting the User or multiple User faces and detecting audio. The videos being played all belong to playlists that automatically switch based on the number of faces captured by the camera. In case the User gets bored of the video being shown then they can skip to the next one by producing a sound. The Operator mode is more complex having a graphic user interface and several windows. The mode starts with the video library that shows all the available videos to display to the user. Here the Operator can choose a video to play and watch it in a detail window where metadata is also displayed for them to analyse it. If it's not available then they might request for it to be processed. The last available window is the playlist window where different playlists might be created and where the number of faces that trigger each playlist might be specified. The are also available keyboard shortcuts to move between windows. To compute the several videos metadata, dedicated functions from openFrameworks were used as well as openCV functions to calculate and apply filters. The algorithm used for object detection was the ORB (Oriented FAST and Rotated BRIEF) algorithm [6].

VI. Conclusions

It is clear that video processing requires intense computation and it is therefore very time consuming, meaning that efficiency and effectiveness are key points in the area of Computer Vision and should not be overlooked. Regardless, overall the project allowed us to learn a lot about many different computer vision concepts, different multimedia files and applications, algorithms and techniques, dominant problems in the field as well as a brief introduction to unsupervised learning by extracting descriptors from unstructured data. In the future several improvements could be done to the performance like the addiction of multithreading for face detection and little tweaks to the application's graphic user interface in order to improve the overall user experience. Ideally one would even remove the "Semi" from its title in order to make it fully automatic.

References

[1] Marie-luce Viaud, Olivier Buisson, Agnes Saulnier, and Clement Guenais. 2010. Video exploration: from multimedia content analysis to interactive visualization. In Proceedings of the 18th ACM international conference on Multimedia (MM '10). ACM, New York, NY, USA, 1311-1314. DOI: https://doi.org/10.1145/1873951.1874209

[2] Zhe Wang, Matthew D. Hoffman, Perry R. Cook, and Kai Li. 2006. VFerret: content-based similarity search tool for continuous archived video. In Proceedings of the 3rd ACM workshop on Continuous archival and retrival of personal experences (CARPE '06). ACM, New York, NY, USA, 19-26. DOI=http://dx.doi.org/10.1145/1178657.1178663

[3] Wolfgang Hürst, Georg Götz, and Philipp Jarvers. 2004. Advanced user interfaces for dynamic video browsing. In Proceedings of the 12th annual ACM international conference on Multimedia (MULTIMEDIA '04). ACM, New York, NY, USA, 742-743. DOI: https://doi.org/10.1145/1027527.1027694

[4] Nuno M. R. Correia. Multimedia Computing, 2019.

[5] Divyansh Dwivedi. Face Detection For Beginners. (27th April, 2018) Available at: https://towardsdatascience.com/face-detection-for-beginners-e58e8f21aad9 [Accessed 18 June 2019]

[6] ORB (Oriented FAST and Rotated BRIEF), OpenCV 3.0.0-dev documentation. Available at: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_orb/py_orb.html [Accessed 18 June 2019]