Art Wall - Multimedia Computing Project

Ariadna Cortés 64108

1. Introduction

The aim of the project is to build a system to control the interaction, visualization and display of artistic content. While this content will be related through the metadata extracted from the art itself. There is a growing interest in this area, that we have studied taking a look at previous examples to later on implement the best features in our system.
This project will be done using Open Frameworks (OF) that is an open source C++ toolkit, we will also mainly use the OpenCV library. OpenCV is an open source computer vision and machine learning software library.

2. Related work

We have studied three articles, which talk about the two main legs of our software, the user interaction interface and metadata extraction. The first two ones are based on this last one, while the third is mainly focused on the user experience.
The first article and the most similar to our project is ConTour [1] a system developed by Davenport, G. and Murtaugh, M, it is a graphical representation application for narrative purposes for an end user.
The system works by getting a single input as a text file describing the art characteristics. Each file has a keyword, a video clip or a picture and a screen position. So that ConTour reads the database, creates a thumbnail, and displays each item in its location.
An important fact about ConTour is that every keyword has an activation value, when the keyword is clicked the element's activation value is raised, so the system keeps track of the user activity. When a keyword is activated, it activates the items described by that keyword, and they come in front of others. So that the user can clearly see the information that he is looking for. Also, the user is able to modify the interface, by dragging and dropping the elements.

The second paper of study is ShapeCompare [2], developed by Université Paris-Saclay, its goal is to create a shared interactive workspace, in a wall-sized display, where non-CAD* experts could generate multiple design alternatives and collaboratively explore them. The user interaction is a field that we really should take care of, at each step we have to remember to keep it easy and intuitive. Also being able to modify the files is an interesting option, that we could have applied by zooming in and out, to get more definition.

After taking a look at ConTour and ShapeCompare, two softwares about relating multimedia information with metadata, we will take a more application approach with the third paper.

The Mimar Sinan Fine Arts University, talks in their paper [3] about the different uses that technology can have inside the museums. The art piece can be interacted using two ways. The passive interaction in which the user does not have to have any direct action to receive feedback e.g. digital screens. On the other hand the active interaction user has to do an action to receive feedback e.g. touch screens, quick response (QR).

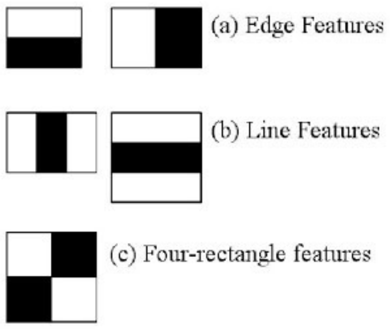
3. Algorithms and techniques

To extract the metadata from the images we extracted their characteristics using different algorithms worked on in class.

3.1 Number of faces appearing in the image or video:

For this implementation hem utilizat l'exemple del OF,opencvHaarFinderExample.

Once we have the image, in grayscale, we use a mask to calculate the filtered image, different masks can be used, see the image contiguous .Each feature results in a single value which is calculated by subtracting the sum of pixels under white rectangle from the sum of pixels under black rectangle.



These features are then used (and selected) in a cascade of classifiers. Selected features are included if they can perform better than random guessing.

3.2 Edge distribution:

To find the edge distribution of the images, we have to apply edge detection methods. The edges are the part of the image that represents the boundary of an object in the image. Also, the pixel values around the edge show if it’s a gradual difference or a sudden change in the pixel values.

The algorithm works as follows, we take the image as a set of values, we have to look for sudden changes in values throughout the image, there will be the edges. Iteratively we are analyzing fragments (3x3 pixels), but we will not do this task manually, so to check if the change is significant we use arrays called kernels.

For this practice we used 3 different kernels, Laplacian, Roberts and Sobel. I found it interesting to use these three, to get more metadata information and because each has a peculiarity when it comes to acting.

Once we have selected a kernel, for example Sobel:

$$h1(x) = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad h2(x) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

We transform the original image to grayscale to make it easier to analyze. Next, we convolute image I with each of the kernels separately so that: I * h1 (x) = I1 and I * h2 (x) = I2. The current result is two images, one with the vertical edges detected and the other with the horizontal ones. Finally, we add the two results and get our image I filtered.

3.3 Texture characteristics:

The approach is the same as for edge detection. But this time the kernel will be Gabor Filters.

$$\int I(x_1, y_1) * g_{m\theta}(x - x_1, y - y_1)\, dx_1 dy_1 = W_{m\theta}(x, y)$$

The way to calculate this gabor filter is as follows, where the value depends on different values, sigma, gamma, theta and lambda:

$$G(x,y) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi \frac{x'}{\lambda}\right)$$

$$y' = -x\sin\theta + y\cos\theta \;;\; x' = x\cos\theta + y\sin\theta$$

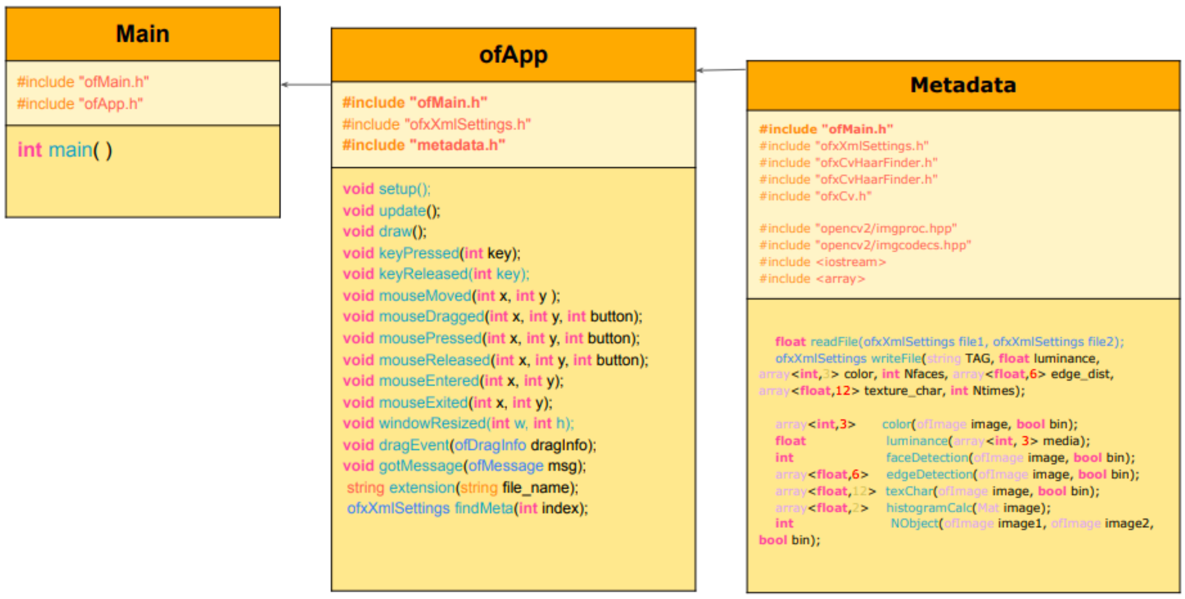
I calculated the value for 6 orientations and 4 frequencies, so I got 6 images results. Then I added them to get the final image result. Finally I extract the mean and variance of those to keep that information as the metadata of the image.

3.4 Number of times a specific object:

First of all we define the keypoints, and the detector type ORB is fast in the execution but lossy. So after that we just have to detect the key point in the image, and find the descriptors. The method makes sure that the output keypoints and descriptors are consistent with each other. Now we have the descriptors for both images,and we just have to match them.We create a matcher that for each descriptor from image 2 does exhaustive search for the nearest descriptor in image 1 using Euclidean metric.

4. Class description

The framework has three classes, the Main class, the App class and the Metadata. Following the diagram below:



```
#include "ofMain.h"
#include "ofApp.h"

int main( )
```

```
#include "ofMain.h"
#include "ofxXmlSettings.h"
#include "metadata.h"

void setup();
void update();
void draw();
void keyPressed(int key);
void keyReleased(int key);
void mouseMoved(int x, int y );
void mouseDragged(int x, int y, int button);
void mousePressed(int x, int y, int button);
void mouseReleased(int x, int y, int button);
void mouseEntered(int x, int y);
void mouseExited(int x, int y);
void windowResized(int w, int h);
void dragEvent(ofDragInfo dragInfo);
void gotMessage(ofMessage msg);
 string extension(string file_name);
 ofxXmlSettings findMeta(int index);
```

```
#include "ofMain.h"
#include "ofxXmlSettings.h"
#include "ofxCvHaarFinder.h"
#include "ofxCvHaarFinder.h"
#include "ofxCv.h"

#include "opencv2/imgproc.hpp"
#include "opencv2/imgcodecs.hpp"
#include <iostream>
#include <array>

   float readFile(ofxXmlSettings file1, ofxXmlSettings file2);
   ofxXmlSettings writeFile(string TAG, float luminance,
array<int,3> color, int Nfaces, array<float,6> edge_dist,
array<float,12> texture_char, int Ntimes);

   array<int,3>      color(ofImage image, bool bin);
   float             luminance(array<int, 3> media);
   int               faceDetection(ofImage image, bool bin);
   array<float,6>    edgeDetection(ofImage image, bool bin);
   array<float,12>   texChar(ofImage image, bool bin);
   array<float,2>    histogramCalc(Mat image);
   int                NObject(ofImage image1, ofImage image2,
bool bin);
```

The initial class is the Main, which initializes the window and creates an instance ofApp.

So let's move on to the ofApp class, this one has 3 main functions: *setup()*, *update()*, *draw()*. The *setup()* initializes the images by loading them into the directory, it also initializes the variables needed to later calculate the motion detection. In order for the camera to be updated on each frame and to be able to find its contours, we use the *update()* function. Finally, the *draw()* function will be responsible for printing things on the screen. We will show at all times the image we are analyzing and we have two options, show the rest of the photos according to their similarity in metadata or show the camera for the interactive part of the system.

In order to calculate the metadata, I created a separate function that reads the metadata of each image and saves it in an XML file that will be written and later on readed with metadata functions writeFile and readFile.

Ultimately, the Metadata class has two functionalities. The first is to read and write XML file, as we mentioned before and the other functionality is to calculate the metadata of the images, to do this we have the following functions:

- *color() :* It will return the average of each of the RGB values in the image.
- *luminance():* With the values calculated in the previous function, we calculate the luminance with the formula seen in class, Y = 0.2125 * R + 0.7154 * G + 0.07121 * B.
- *faceDetection() :* Applying the algorithm explained above 3.1 it will return the number of faces found in the image.

- *edgeDetection() :* We look for the edges with three different operators in order to get more metadata, and return the mean and variance of the Laplacian, Roberts and Sobel operators.
- *texChar() :* We apply the algorithm as described 3.3, in this case it will return the mean and the variance of the image after applying the filter.
- *histogramCalc() :* It is responsible for calculating the mean and variance of matrices. Extracted form [4]
- *NObject() :* Compare two pictures and find the matching objects.

5. Conclusions

The aim of the project was to build a system of control, interaction and interrelated art display, which can be displayed on a wall display. It had to contain image and video sorting the content according to the metadata.

The challenges I encountered above all:

- OpenCv: there were functions that were read well at first and others that it did not understand despite being, in theory, part of the same include. Some needed the prefix *cv::* others did not. This has been a recurring issue in my program.
- Video introduction: All functions have the possibility to add video implemented and commented.I don't understand why, the program breaks every time I put a video on.
- Performance issues: The program is separated into two modes (main mode and camera mode) for proper performance. When I put the camera in main mode it worked in very few frames, making it useless.

It has been the most ambitious project that I have faced so far in my university career. I had no knowledge of motion recognition, data extraction or programming in Xcode and I know that I did not fill in all the required fields, without the video. But I have learned a lot both theoretically and in terms of programming. It has been a great growth for me to have to deal with all the problems by myself.

7. References

[1] Davenport, G., & Murtaugh, M. (n.d.). *ConTour*. Http://Alumni.Media.Mit.Edu/. Retrieved April 21, 2022, from http://alumni.media.mit.edu/~murtaugh/thesis/ConTour/ConTour.html

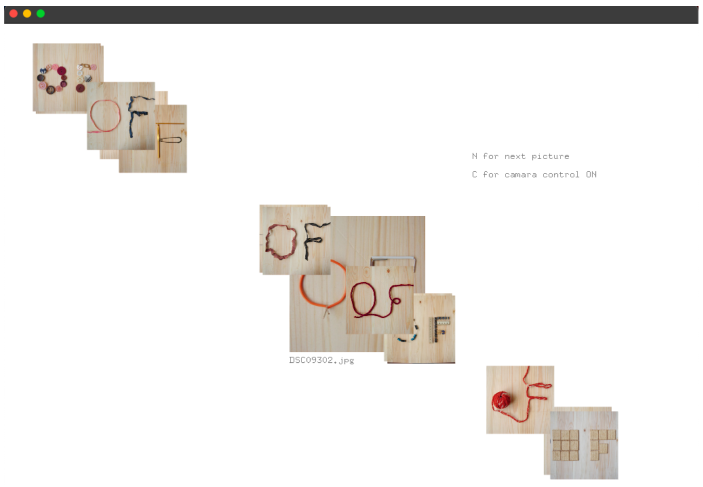
[2] Okuya, Y., Gladin, O., Ladevèze, N., Fleury, C., & Bourdot, P. (2020, April). *Investigating Collaborative Exploration of Design Alternatives on a Wall-Sized Display*. Université Paris-Saclay, CNRS. https://doi.org/10.1145/3313831.3376736

[3] Gamze, E., & Arabacioglu, B. (2020, October). *DIGITAL INTERACTIVE EXPERIENCES IN CONTEMPORARY ART MUSEUMS.* The Turkish Online Journal of Design Art and Communication. https://doi.org/10.7456/11004100/007.

[4]*Histogram Calculation — OpenCV Documentation*. (s. f.). OpenCV Documentation. Recuperado 9 de junio de 2022, de https://vovkos.github.io/doxyrest-showcase/opencv/sphinx_rtd_theme/page_tutorial_histogram_calculation.html

Appendix: User manual

The main page is presented as shown in the figure. In this we can see the image which is being analyzed in the center of the window. Surrounding it are the other images contained in the directory, sorted by proximity according to the similarity/difference found when calculating the metadata of each of the photographs.

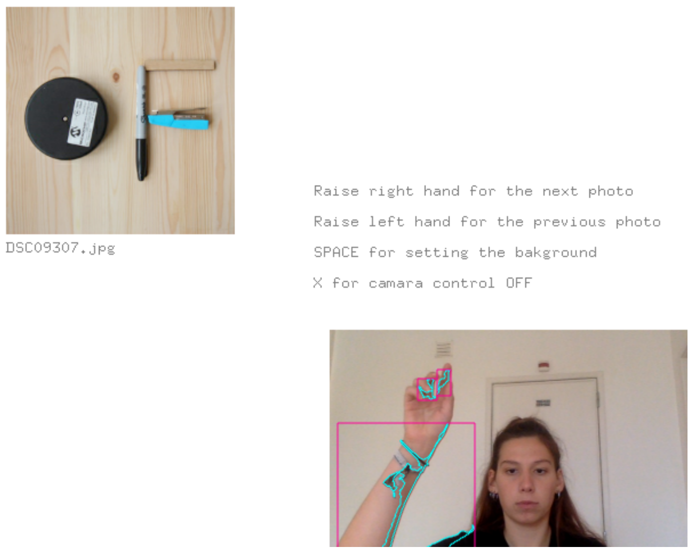


In this window you can:

- Press the **N** key to move on to the next image and see how it relates to the other images in the directory.
- Press **C** to activate the *camera mode*.

The *camera mode* has the image in the center and the camera on one side. The contour detector is activated, so if you raise your right hand it will move to the next image and if you raise your left hand it will move to the previous image. Also:

- Pressing the **SPACE** will recalculate the background, so that the contour detection works correctly
- By pressing the **X** key the user will be able to return to the main mode.


DSC09307.jpg
Raise right hand for the next photo
Raise left hand for the previous photo
SPACE for setting the bakground
X for camara control OFF