Introduction to Image and Video Processing

COT 4903

Pablo Adell Álvarez-Rico

Z23440900

Introduction to Image and Video Processing
COT 4930

Assignment 3

1. Overview

For this project one was asked to be creative, without any specific focus. In my case I used a visual recognition library called OpenCV to do face tracking to detect faces within a picture.

This problem is approached passing a image through a detector that has been pretrained with a workload of samples to be efficient.

```
if(original.empty()) printf("Error abriendo el archivo\n");
  printf("Testing OpenCV\n");
  printf("Loading Haar-Cascade\n");
  CascadeClassifier faceDetector;
  bool loaded1 =
faceDetector.load("data/haarcascade_frontalface_default.xml");
  printf("Training Loaded? %d\n", loaded1 );
```

However, to execute this method the user is given a UI so it can select whatever image he or she wants. After both have been selected, a button shows up to start the execution.

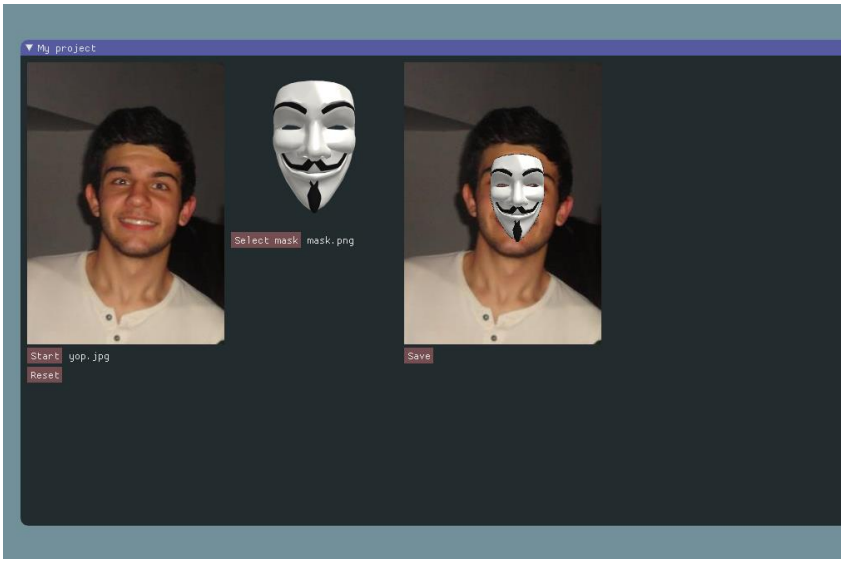
After the button is clicked, the algorithm checks if the size of the mask to be applied can fit within the area defined for the face given by OpenCV after going through the face detection routine.
If the mask doesn't fit, it'll be send to another method made to resize the picture specifically for the face's size. This method was given by TA Esad Akar and can be found on his github.

```
resize_image(&b, faces[0].width, faces[0].width );
```

Once the mask has been resized, if needed, the program goes through two for loops to iterate through the pixels that form the face replacing them by the mask's pixels. However, as the mask

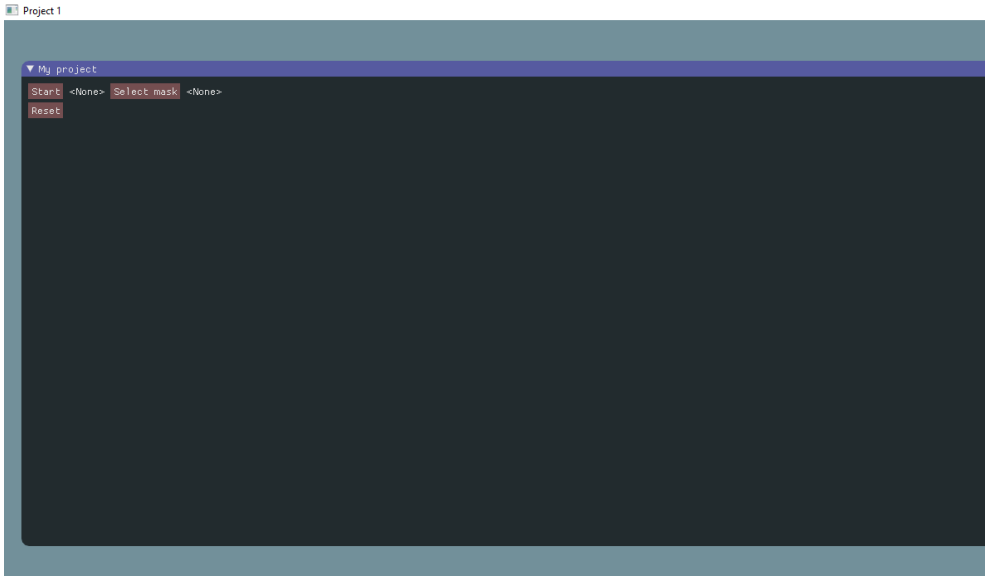
must have transparent background to give optimal performance, whenever it detects 0 or that alpha is 0, it replaces that pixel component by the original one and so, the result is improved.

```
for(int x = faces[0].y; x < faces[0].y + b.height; x++){
      for(int y = faces[0].x*4 ; y < (faces[0].x + b.width)*4 ; y++){


              if(b.pixels[j] ==0){


                pixels[(a.width * 4 * x) + y] = pixels[(a.width * 4 * x)
+ y] ;
              }else{
             pixels[(a.width * 4 * x) + y] =(b.pixels[j]);
}

      j++;


    }

  }
j = 0;

}
```

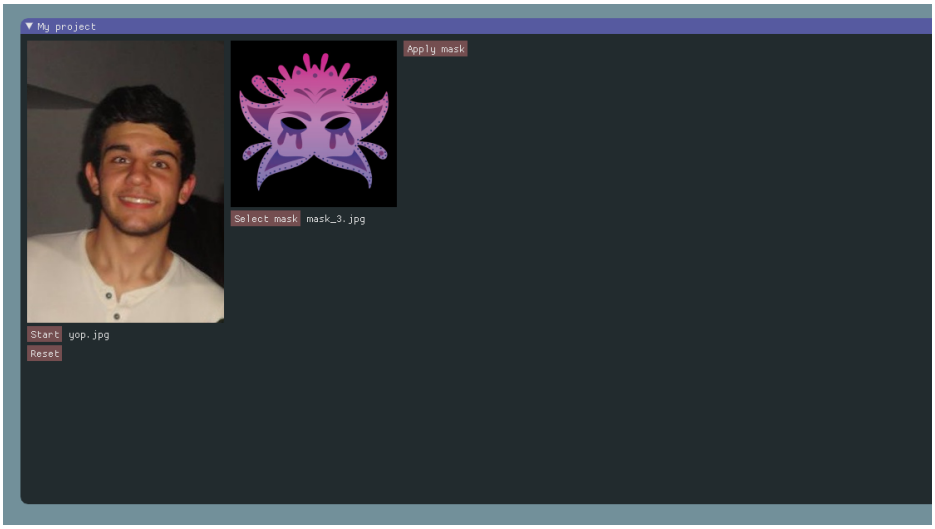


2. Results

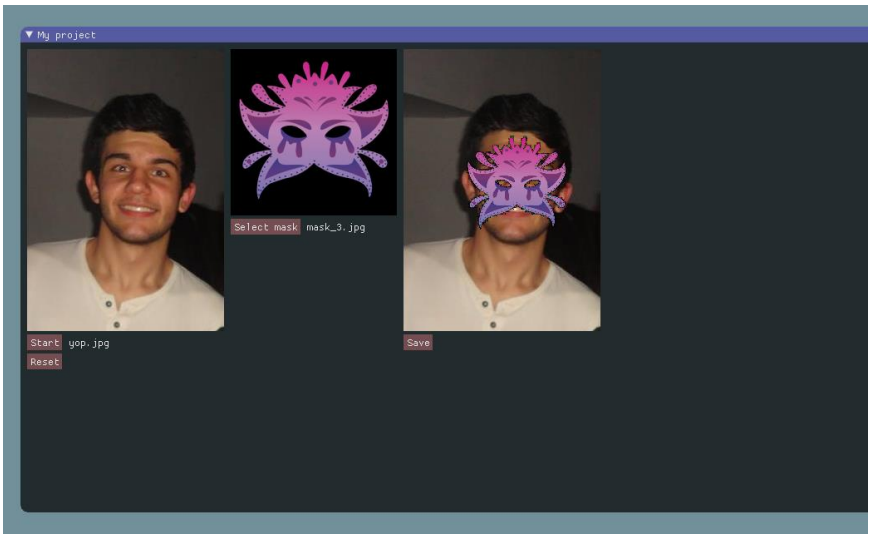
In this section the results of the program running will be shown. As an example, it will be provided a screenshot of the program's UI.



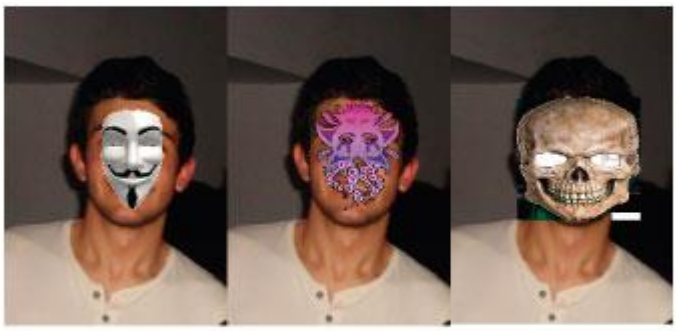
This is the initial UI provided to the user before any file has been selected. After that, the user will select a mask and a picture to be manipulated in order to be able to execute the program



After that, once the “Apply mask” button is clicked, the program starts working and displays the result once it is done.



The program works with a large variety of masks, some examples will be given next.



Results after applying three different mask over the same subject

3. Algorithm

The algorithm is quite simple and OpenCV is its pinnacle. At first, when the user selects the picture, before displaying it, we create a instance of a Mat object as we need it to be able to execute the face tracking over the subject we want. The after the subject face has been detected, we set a squared area around it using the initial x and y pointers to iterate through the array that will replace the original face by the selected mask. Once this is done, we load the texture on a new image and display it so the user can see the result.