Face Mapping

1. Aditya Prakash
2. Pranav Gujarathi
3. Shobhit Mittal
4. Shubham Kumar
5. Tarannum Khan

Abstract

This project describes a method for eye and mouth detection and eye center and mouth corner localization, based on geometrical information is presented. First, a face detector is applied to detect the facial region, and the edge map of this region is extracted. A vector pointing to the closest edge pixel is then assigned to every pixel. x and y components of these vectors are used to detect the eyes and mouth. For eye center localization, intensity information is used, after removing unwanted effects, such as light reflections. For the detection of the mouth corners, the hue channel of the lip area is used.

Acknowledgement

This proposal discusses the detailed study we went through while working upon this project. This project was carried under the aegis of Electronic Section, Indian Institute of Technology, Roorkee.

First and foremost, We would like to thank our esteemed staff advisor, Mr. Kamal Singh Gotyan for his consistent guidance. He has been a source of inspiration and motivation throughout the project work. Then, we would like to thanks our hardworking mentors, Padmanabh Pande for his firm support and great ideas, Rahul Ratan Mirdha who always stood by our side and supported us like a pillar at each step of our project and Gaurav Waghmare for his help and coordination with us.

Finally thanks to our friends and family who for their help ,support and motivation.

Introduction

Facial feature localization is an essential task for a number of important applications like face recognition, human-computer interaction, facial expression recognition, surveillance, virtual makeup etc. A multi-stage approach is used to locate features on a face. We worked upon visual studio integrating with cmake. We wrote code in cpp to automatically replace facial features on an image of a face with various type of masks using dlib software in real time.

While sophisticated MRI and X-Ray machines leave little to the imagination, they haven’t been around that long. Human disease and disorders, however , precede them by millennia.

Techniques can be used today to read the skin on our face to help determine what’s causing flare-ups or blemishes using Face Mapping to connect a point on your face to an organ or body part so to know what to treat internally for clear external results. Face Mapping can be used to analyse the skin to provide insight into skin’s past and present and allows the subsequent prescription to a targeted regime. A professional skin Therapist uses Face Mapping to divide the facial landscape into fourteen zones. Each zone is examined to determine where conditions such as congestion, breakouts, dehydration, and hyper-pigmentation are present.

Content

1. **Face detection**

First step was to detect the facial landmarks. We used dlib to extract facial landmarks which is a general purpose cross platform software a general purpose cross platform software a general purpose cross platform software written in the programming language C++. Its design is heavily influenced by ideas from design by contract and component based software engineering. Thus it is, first and foremost, a set of independent software components. It is an open-source software released under a Boost software License.

We implemented the algorithm used in the paper One Millisecond Face Alignment with an Ensemble of Regression Trees used through dlib. The algorithm itself is very complex, but dlib’s interface for using t made it simple.

We take images in sequence by reading the camera in the form of a numpy array, and return a 68x2 element matrix, each row of which corresponding with the x,y coordinates of a particular feature point in the input image. The feature extractor requires a rough bounding box as input to the algorithm. This is provided by a traditional face detector which returns a list of rectangles, each of which corresponding with a face in the image. To make the predictor a pre-trained model is required which has been made available in the dlib software.

We find the frontal faces in an image and estimate their pose. The pose takes the form of 68 landmarks. These are points on the face such as corners of the mouth, along the eyebrows, on the eyes and so forth. The face detector is the fastest when the compiled with at least SSE2 instructions enabled.

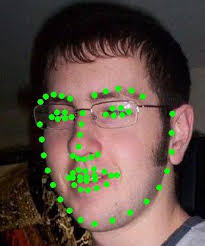
The face detector is made using the now classic Histogram of Oriented Gradients(HOG) feature combined with a linear classifier, an image pyramid, and a sliding window detection scheme. The object detector is fairly addition of human faces. Various machine learning tools have been used to create dlib’s face detector.

1. **Translate, Scaling and Rotation**

Once we have our two landmark matrices, each row having coordinates to a particular facial feature. For eg the 30th row gives the coordinates of the tip of the nose. Then, we translate by subtracting the centroid from each datasets and divide each points by its standard deviation to scale the points of the vector of the mask such that they fit as closely as possible to the points in the vector of the face, the idea being that the same transformation can be used to overlay the mask on the frame of the camera while processing. Then, we rotate using singular value decomposition. Ordinary Procrustes Analysis can be used for much greater efficiency.

1. **Colour Correction**

Then, we colour correct the mask for proper matching. It is done by dividing the Gaussian blur of mask and then multiplying by the Gaussian blur of face. We are using a RGB scaling colour-correction, but instead of a constant scale factor all of the mask, each pixel has its own localised scale factor to account for the difference in lightning.



68 extracted points as indicated in this picture.

Conclusions

Our project has been designed for detecting faces in real time and superimposing different kind of masks on the face. We have used visual studio and integrated it with the dlib software and used the C++ language for the implementation of code.

Limitations

Using visual studio has led to the slower processing speed of superimposition of mask and adding video effects in real time which could be improved to some extent by using some other high level software. Other than this, we have faced some problem due to the use of the normal projectors leading to not fully 3D effects implementation which could be achieved by using higher standard projectors.

Future Scope

1. The algorithm can be developed further used in face mapping can be used to turn 2D image into a simulated 3D model of a person’s face. In a single second, it can turn an identifiable partial snapshot into a very identifiable headshot can be used as a better law enforcement tool.
2. Facial recognition can be implemented into a higher level by converting it into a full body recognition which could be used for security and surveillance purposes.
3. Face scan for smartphones by using the face unlock features to allow to unlock android phones using faceprint that is a map of the unique structure of the face .

References

1. <http://dlib.net>
2. <https://matthewearl.github.io/2015/07/28/switching-eds-with-python/>
3. <http://docs.opencv.org/2.4/doc/tutorials/tutorials.html>