Face detection in video and adding Snapchat filters

Bojana Ivanovic, Vanja Mijatov

Faculty of Technical Sciences, Novi Sad

Problem description

Face detection in computer vision has become extremely popular, useful and the basis of solving many problems in this field. As such is used for different needs and purposes. Considering that face detection has become a huge part of the social networks that occupy a large part of human lives, we wanted to create something similar to well known application called Snapchat. Our idea was to enable processing videos (that are pre-existing on user's computer or recorded with camera) by detecting faces and adding desired Snapchat alike filters to detected faces.

Conceptual solution and goals

Solution consists of three greater units:

- **Upload video and extract frames** (10 frames per second for recording video and for other uploaded same number of frame per second as input video)
- **Face detection on frames and parts of those faces** (using *dlib* library - **HOG+SVM** algorithm and 68 significant points of the face and *OpenCV* library - **HAAR** cascade classifier)
- **Add the selected filter to the recognized region**

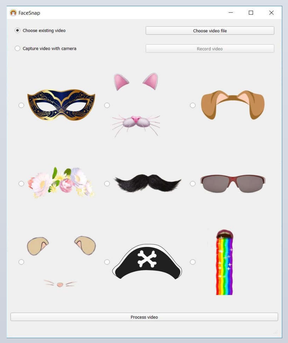

***Figure 1**. Dashboard of the application.*

Our main goal was to try detecting faces using two common approaches used in computer vision:

- **Histogram of Oriented Gradients**
- **Haar Cascades (based on Viola-Jones method)**

and comparing their results.
Then, using one of these approaches, facial landmarks were tried to be detected and filters attached to images in order to cover parts of face(s) and this operation to be evaluated.

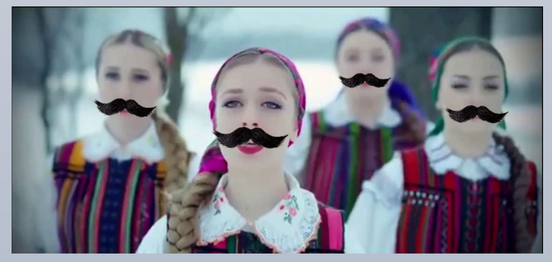

***Figure 2**. Example of one frame from the output video where four faces were detected even though only one is in the foreground and the others are blurred.*

Face detection - algorithms

HOG (Histogram of Oriented Gradients)

HOG is a feature descriptor used to detect objects in computer vision and image processing. The HOG descriptor technique counts occurrences of gradient orientation in localized portions of an image - detection window, or region of interest.

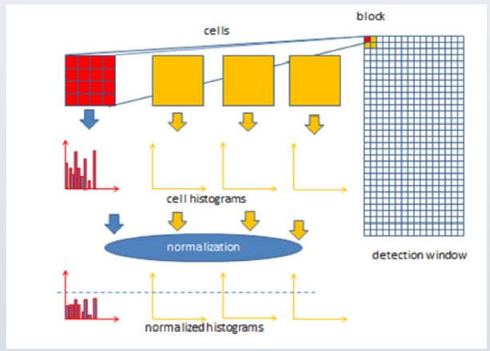

***Figure 3**. The way HOG descriptor works.*

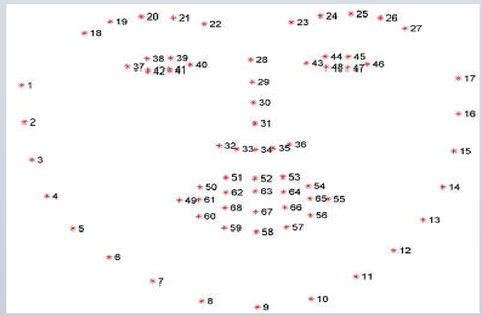

***Figure 4**. 68 significant points on the face.*

Haar Cascades

Haar is a classifier which is used to detect the object. Algorithm provides rectangles where faces might be and after that specialized Haar Classifiers are used to find face parts.
With that information it is relatively easy to put the stickers in adequate face part in the image.

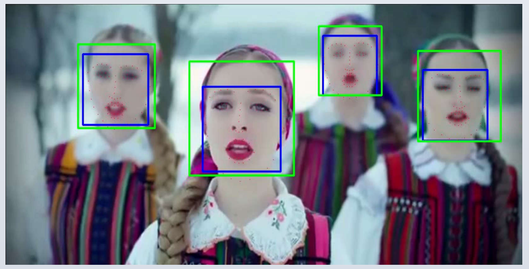

***Figure 5**. Rectangles represent bounding boxes around detected faces.*

At first, in combination with HOG, we tried to use shape predictor with 5 significant points of the face and it was simple but not so efficient.
As a better solution for our problem was shape descriptor with 68 significant points and it enabled us easier way to put desired sticker on adequate face part.

Results

Dlib vs OpenCV

Algorithms implemented by dlib and OpenCV have different approaches and their success in detections is much different.

1. Accuracy

After evaluating results generated based on testing done on different samples (with different number of faces, different brightness, with faces in motion) it is concluded that:

- **dlib is much accurate when detecting faces than OpenCV**
- **dlib has success ratio over 90% which is significantly higher that OpenCV's approach which showed average results around 70%**

Accuracy of dlib allowed better detecting face parts and therefore more accurate covering face parts with filters.

2. Processing time

- **processing time is much higher for dlib.**

After testing on different machines, it was concluded that *dlib required sometimes even more than 100 times more time for every frame* while OpenCV was so fast that if was almost real time.

Evaluation

For evaluation of how successful was covering of part(s) of face(s) with filters we used ***IoU* (Intersection over Union).** IoU is an evaluation metric used to measure the accuracy of an object detector on a particular dataset. In our case, we used IoU to measure to what extent some filter covers certain face part.

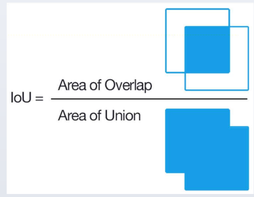

***Figure 6**. Computing the IoU*

Besides IoU, we compared success of the detection of exact number of faces in each frame for both dlib and OpenCV and it is being shown in command line as well as percentage of IoU.

Data set

To demonstrate our solution we created small data set that contains different types of videos like videos with one person in it, more that one person, videos with different levels of brightness, video containing some of the actors in the foreground and the rest of them in the background - blurred, etc. Videos from data set have different frame sizes, frequency pre second, quality and could have different format.

Conclusions

HOG based algorithm implementation showed much higher accuracy and could be used for more sophisticated purpose. On the other hand, it is really slow and could not be used for real time detecting. Haar Cascade based implementation has lower accuracy but much better performance. This approach could be used for real time detection and systems that do not require much precision.

***Figure 7**. Figure shows frame before adding filter on it.*

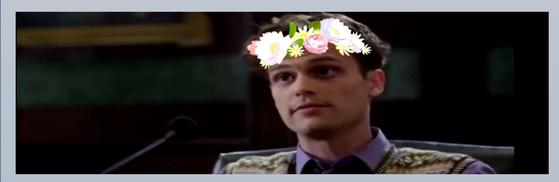

***Figure 8**. Figure shows frame after adding filter on it.*