Face Obfuscation

Trevor Nouvel
Clemson University - CPSC 4820

ABSTRACT

Facial recognition is a concept first pioneered in the 1960s . Woody Bledsoe, Helen Chan Wolf and Charles Bisson attempted to detect human faces with computers, however they were severely limited by the technology at the times [1]. Since then, the technology has evolved substantially and is used in a variety of applications. For example,

Law enforcement agencies use it as a way to identify suspects or other persons at the scene of a crime. Most mobile phone cameras utilize facial recognition to focus on faces to capture better portrait images. Social media companies are known to use it to identify users who's faces appear across multiple posts. Facebook has received significant backlash as it could present a breach in privacy.

In today's world, many applications present a dichotomy of privacy and usefulness to society as a whole. Many still feel that privacy is the most important aspect to one's life, however.

KEYWORDS

fogging, censorship, face detection, obfuscation

1 INTRODUCTION

Fogging is a way to censor an object from view in a frame [2]. It essentially blurs out a a section of a video or photo. This can be seen in live-action television that blurs out faces or company logos. Production crews blur out faces of persons who do not give consent to be on camera. In addition, they will blur out company logos. The reasoning is to avoid lawsuits as the film could paint a person or company in a bad light.

It is important to allow people to blur themselves from an image or video. The photo or video can keep its integrity as well as the person who may incidentally appear in it.

2 BACKGROUND AND RELATED WORK

In Section 1, it was mentioned that facial recognition has come a long way. Modern programming languages can utilize libraries to assist this process. A popular library is OpenCV. Originally written in C/C++, it can be imported into Python as well [3]. Developers have access to a multitude of functions that can be manipulated to work to a desired effect. For example, the sensitivity of detection and facial feature to be detected can be changed.

Cascades. OpenCV uses Haar cascade filters. Created by positive samples of a region of interest (ROI), the cascades are trained to identify certain properties of the face [4].

2.1 Overview of the design

There are several methods to censor a face from a frame. In Figure 1, a Gaussian blur is applied to the woman's face. Gaussian blur is meant to reduce noise and detail of an image [5]. Another method of censorship is to pixelate the face.

Figure 1: Facial Fogging

3 METHODOLOGY/DESIGN

Face obfuscation requires a face(s) to be detected then censored. This requires several steps to properly implement:

First, all faces must be found. Videos will require to be read frame by frame while tracking each face. OpenCV offers the DNN Module (Deep Neural Networking). The module allows a user to load different models for the deep learning [6]. One useful framework comes from Berkeley AI Research (BAIR) called Caffe [7]. It takes in a two files: *.prototxt and .caffemodel. The prototxt defines the architecture of the model, and the caffemodel holds the weights for the layers [8]. With the help of the DNN, faces are detected quickly and effectively.

Users are given an option to censor faces on an image (*.jpg) or video (*.mp4) in a window. After selection, the user will be prompted for obfuscation by blur or pixelation. The program also allows the user to choose the level of censorship by factors.

3.1 Gaussian Blur

A Gaussian filter is executed by convolution of each point in the region of interest with a Gaussian kernel [9]. The sum of the convolutions is returned to the frame over the face. Thus, the face is blurred. OpenCV has implemented this function.

$$G_0(x, y) = Ae^{\frac{-(x-\mu_x)^2}{2\sigma_x^2} + \frac{-(y-\mu_y)^2}{2\sigma_y^2}}$$

Figure 2: 2D Gaussian Blur Formula

3.2 Pixelation

In order to pixelate the face, the region of interest must be determined. The user defines a number of "blocks" to be displays over the face. Then, the mean RGB values of each block are calculated and placed in a rectangle. This occurs in the x and y directions.

```
(b, g, r) = [int(x) for x in cv2.mean(roi)[:3]]
cv2.rectangle(img, (start_X,start_Y), (end_X,end_Y), (b,g,r), -1)
```

Figure 3: Mean RGB and Rectangle

4 RESULTS

Overall, the program worked how expected. Faces were easily detected with the help of the Deep Neural Network. I felt that this was a cleaner way for face detection rather than simply using Haar cascade filters. It also seemed to offer more reliability for head movements.

It is important to find the correct region of interest of the face. This allowed me to easily execute blurring and pixelation. The implemented Gaussian blur function in OpenCV was very helpful as well. However, I had to find a way to pixelate the region of interest which took a little more time.

Unfortunately, I was unable to implement a few features I wished to. Having the ability to choose a specific face to censor was desired. This could be done with a face object class with an update function to "remember" faces. Or, I could train the model myself with a specific face to be censored if detected.

There are many more applications to using DNNs as well. With the accuracy and precision of object detection, one could try overlaying images over the face. Snapchat is a good example of this.

5 CONCLUSION

Overall, this was a satisfying and fun project to implement. Despite some shortcomings in the addition of some features, the program runs as expected and censors all faces on a frame. I thought it was important to give the users options for filetype, choice of file, obfuscation type, and factor of obfuscation.

With so many cameras around the world, it is important to have the ability to remove your face from an unwarranted picture or video. It was very interesting to see how this can be done in code.

6 REFERENCES

[1] A brief history of facial recognition: https://www.nec.co.nz/market-leadership/publications-media/a-brief-history-of-facial-recognition/
[2] Fogging (censorship): https://en.wikipedia.org/wiki/Fogging_(censorship)
[3] Face Recognition with Python: https://realpython.com/face-recognition-with-python/
[4] Cascade Classification: https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html
[5] Gaussian blur: https://en.wikipedia.org/wiki/Gaussian_blur
[6] OpenCV DNN Module: https://learnopencv.com/deep-learning-with-opencvs-dnn-module-a-definitive-guide
[7] Caffe Framework: https://caffe.berkeleyvision.org
[8] Face Detection with OpenCV and Deep Learning: https://pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning
[9] Smoothing Images: https://pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning