State of the Project

Forehead tracking with opencv

James MacFadyen – 20/04/2019

2019

Contents

[Haar Cascade Forehead Tracking 2](#_Toc6670824)

[Overview 2](#_Toc6670825)

[User Guide 2](#_Toc6670826)

[Head Pose Estimation using TensorFlow 3](#_Toc6670827)

[Overview 3](#_Toc6670828)

[Issues 3](#_Toc6670829)

# 

# Haar Cascade Forehead Tracking

## Overview

Functionally, the included script works as far as detecting foreheads in a given video file. It uses the haar cascade classifier to detect faces, and then draws a rectangle around the upper third of the face, in order to isolate the forehead. Lack of precision notwithstanding, as a rough and ready method for forehead detection this works. Also currently baked into the script are elements to write a list of coordinates that have been known to have faces in them (with a view to tracking movement and rotation over a long period of time) to a text file, and a similar element to write the BGR pixel values of the region of image being dealt with. This latter functionality needs work as of this document, as it does not indicate the **dominant** colour, but rather the **average** colour, for the given region. This may pose problems relating to measuring green-channel variation.

Issues with this route for automating the detection of green-channel changes in skin colour include a variety of hurdles surrounding the PnP (“Perspective-n-point”) problem, with the haar classifier included in this repository providing only facial recognition for faces looking almost directly at the camera. Depending on use-case for the final project, the licensing for the haar cascade may also present an issue, as it is open to use under the Intel License Agreement.

In the following section, I include a getting-started guide for running the python script on a user’s local machine.

## User Guide

1. The local machine must have either Python 2.x or Python 3.x installed.
2. The local machine must have NumPy and MatPlotLib libraries installed for Python (these can be installed via “pip” – for more information, see (https://pypi.org/project/pip).
3. The local machine must have the openCV2 library installed. Note: there exist common libraries for C++, so it is important to ensure that the Python version of OpenCV is downloaded and installed.
4. This repository must be cloned somewhere convenient. This can be done with Git – for more information, see (https://git-scm.com/).
5. Next, in the cloned repository, the “faceRecognition.py” file must be modified on line 8, with the path being replaced with the path to whichever piece of media you wish to run the script on.
6. Finally, navigate the command prompt (on Windows 10 this can be accessed in the current directory easily by clicking on the folder path just below the “view” button, and typing “cmd” before pressing enter) to the path containing the “faceRecognition.py” file, and type “python faceRecognition.py”, without the quotes. Depending on the way your environment variables/PATH are set up, you may need to type “python3 faceRecognition.py”.
7. The video file should run, and any foreheads facing the camera should be outlined with a rectangle. If there are any false positives, or if foreheads are not adequately detected, consider editing the “detectMultiScale” variables on line 22.
8. Output files “coordinates.txt” and “pixelValues.txt” will be created in the root folder.

# 

# Head Pose Estimation using TensorFlow

## Overview

My current work on the project has been in aid of utilizing the robust head pose estimation found here: <https://github.com/yinguobing/head-pose-estimation>. As a cursory glance at the included readMe will demonstrate, this is an established implementation of the head pose tracking that was discussed for this project. In order to run the script, first it is necessary to download and install TensorFlow, which is Google’s machine learning library. Currently, my home use and work computers both run Windows 10, and initially this posed a problem as the build of TensorFlow used with this project appears to run well only on Linux. The author indicates that they have also tested their Head Pose Estimation on Ubuntu. As such, I opted to test the script in an Ubuntu virtual machine (running through VirtualBox). After installing the requisite software on my laptop, it became clear that my CPU was not being flagged for “AVX” (Advanced Vector Extensions) despite being physically capable. This is an issue, as TensorFlow appears to require AVX in order to run with this project. I then performed the same setup on my desktop machine, only to face the same issue. The next step for implementation would be to acquire a physical machine running Ubuntu 16.04 LTSE, or attempt to run it through a different virtual machine (this seems unlikely to resolve the issue).

## Issues

This implementation would be even more black-box than the already oblique haar cascade implementation. It has the massive benefit of having already been designed and developed, and using it would present a much faster development. Manually crafting software that solves PnP seems, after extensive research, well beyond the scope of this project – and so it maybe necessary to utilise a method such as this. A big issue involved in using this head pose estimation would be the inclusion of various licenses, which can be reached via the readMe. For academic or commercial use, it might be difficult.

It may also be harder to adapt a pre-baked project to any specific needs, particularly with a view to automating drones in order to record forehead movement. Depending on just how “black box” the software is, it may actually be easier to adopt the principles from the code, and reverse engineer something specifically relevant to the project.