Face Detection and Tracking

Project Name: Face Detection and Tracking
Project Team: Shain Mayville
Project Manager: Dr. Imran Ahmad

Introduction

Detection and tracking can be done in several different ways, as well as several applications. Combined they can form a powerful tool in many real-life applications. Some applications include self-driving cars, computer interaction, or virtual reality. This project introduces concepts and open-source libraries for programming developers to use in order to achieve detection and tracking faces while using a camera.

This report will discuss the process of learning and creating detection and tracking faces within a video. This includes the goals, steps taken in the process, some problems encountered, and how to build the application using the C++ code file.

Goals & Take Away

In order to detect and track faces from a webcam it is important for the programmer to understand the concepts involved. The project required me to learn OpenCV, and 2 compatible languages; Python and C++. The project also taught me how to setup the respectful environments, improve my problem-solving skills, and manipulate images using OpenCV.

Steps Taken

Although face detection and tracking are the main concerns regarding this project, there are several steps leading up to this. The project has introduced me to Python, C++ and OpenCV. In order to understand and utilize the different functions available in OpenCV I had followed a tutorial introducing different concepts of OpenCV using Python.

The tutorial I followed initially is called "OpenCV for Python Developers". Recorded by Patrick W. Crawford, the tutorial first teaches the student to setup a python environment with OpenCV and numpy. Once this is completed, he continues on with the manipulation of images using OpenCV. This includes concepts like pixel filtering, color channels and drawing on images. He then furthers the usefulness by introducing the ability to read from a webcam.

The main focus by the end of the tutorial is object detection. Crawford introduces object detection using a few different methods. For instance, color detection, contour detection, and Canny detection were some of the examples Crawford uses to detect objects. The detection method used to detect faces in his tutorial is called haar cascading. This uses a file to find a certain type of object. Although this is not limited to faces, face detection is the example given by the end of the tutorial.

I decided the next logical step for the project would be to develop a program that detected faces from a webcam. Since input from a webcam and face detection were both introduced, I applied these concepts together. The end result was very simply a program that drew a purple box around any detected face in each frame shown from the webcam.

Once I had learned OpenCV in Python, I had to then learn how to program in C++. I started by following the tutorial "Install OpenCV 3 on Windows". The tutorial goes through how to setup a C++ (and Python) OpenCV 3+ environment. After the setup was complete, I then had to rewrite my program in C++. As I did not know the syntax for this language, I skipped ahead, and looked at a tutorial for object tracking in C++ using OpenCV. The example I found shows the code in order to track a moving object in a video from a predefined box in the code.

From reading the example I rewrote my python code using Visual Studio to help me catch any syntax mistakes I made while learning C++. Once the application was translated correctly, I had to implement a tracking component. While looking back at the example for tracking, it was clear that the main task would be passing the detection boxes into trackers.

To finalize the program, I had to find a way to prevent a 'new' face from being detected if the face was already being tracked from a previous frame. To accomplish this, I checked whether the new faces center point was within an already tracked objects box. This proved to be enough to prevent the addition of duplicate trackers for each face detected.

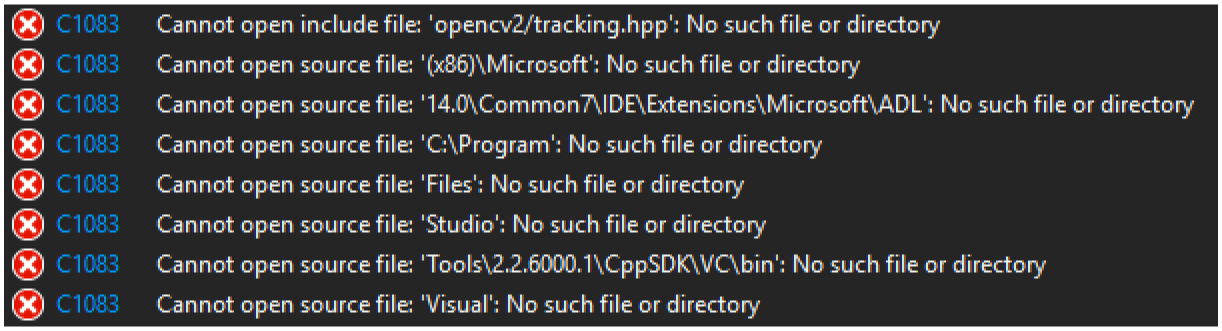
Finally, I added random colours to each tracking box to differentiate between each tracker present. I achieved this by creating a class that held the data (colour, defining box, and tracker) for each detected object to simplify the readability of the program. This also has the added benefit of removing all relevant data when a tracked object is lost.

PROBLEMS AND OBSTACLES

Along the way through the project there were several obstacles I had to overcome. Some were minor, others I spent a lot of time trying to fix. This section will discuss some of these problems, and what I did to overcome them.

The first problem I encountered was the lack of knowledge regarding the subject, as I had never previously used Python, C++ or OpenCV. I first had to extend my knowledge to understand and write code using these programming languages and toolkit. I proceeded by using tutorials and examples to learn the relevant content.

While completing the first half of my work (programming in Python), I did not run into any major issues. Using the information from OpenCV for Python Developers" I was able to write a program to detect a face using a webcam feed without much issues. Rewriting the program in C++ was more of a challenge. For instance, the built-in compiler in visual studio would produce errors every other run. This proved to be a minor inconvenience at the time, as I would just compile the project twice for the time being. From what I can tell by the errors received, the computer was trying to parse (break-up) the directory path on white space. This effectively meant the file could not be found whenever the compiler tried to look for it. I have included an image below demonstrating the errors.



```
C1083    Cannot open include file: 'opencv2/tracking.hpp': No such file or directory
C1083    Cannot open source file: '(x86)\Microsoft': No such file or directory
C1083    Cannot open source file: '14.0\Common7\IDE\Extensions\Microsoft\ADL': No such file or directory
C1083    Cannot open source file: 'C:\Program': No such file or directory
C1083    Cannot open source file: 'Files': No such file or directory
C1083    Cannot open source file: 'Studio': No such file or directory
C1083    Cannot open source file: 'Tools\2.2.6000.1\CppSDK\VC\bin': No such file or directory
C1083    Cannot open source file: 'Visual': No such file or directory
```

While the above problem proved to be more of an annoyance, the extension of the same issue was more serious. In order to use OpenCV 3+ with tracking capabilities, the OpenCV_contrib libraries must be added. When following the tutorial on how to implement them I ran into a similar problem in CMake as the one above. CMake attempts to compile using the same compiler as Visual Studio. This leads to a similar error as the one above. An example of the error is shown below.

```
   c1xx : fatal error C1083: Cannot open source file: 'C:\Program': No such file or directory
[C:\Users\shain\Documents\opencv\opencv-
3.4.3\build\CMakeFiles\CMakeTmp\cmTC_57ba5.vcxproj]

   c1xx : fatal error C1083: Cannot open source file: 'Files': No such file or directory
[C:\Users\shain\Documents\opencv\opencv-
3.4.3\build\CMakeFiles\CMakeTmp\cmTC_57ba5.vcxproj]

   c1xx : fatal error C1083: Cannot open source file: '(x86)\Microsoft': No such file or directory
[C:\Users\shain\Documents\opencv\opencv-
3.4.3\build\CMakeFiles\CMakeTmp\cmTC_57ba5.vcxproj]

   c1xx : fatal error C1083: Cannot open source file: 'Visual': No such file or directory
[C:\Users\shain\Documents\opencv\opencv-
3.4.3\build\CMakeFiles\CMakeTmp\cmTC_57ba5.vcxproj]

   c1xx : fatal error C1083: Cannot open source file: 'Studio': No such file or directory
[C:\Users\shain\Documents\opencv\opencv-
3.4.3\build\CMakeFiles\CMakeTmp\cmTC_57ba5.vcxproj]

   c1xx : fatal error C1083: Cannot open source file: '14.0\Common7\IDE\Extensions\Microsoft\ADL':
No such file or directory [C:\Users\shain\Documents\opencv\opencv-
3.4.3\build\CMakeFiles\CMakeTmp\cmTC_57ba5.vcxproj]

   c1xx : fatal error C1083: Cannot open source file: 'Tools\2.2.6000.1\CppSDK\VC\bin': No such file or
directory [C:\Users\shain\Documents\opencv\opencv-
3.4.3\build\CMakeFiles\CMakeTmp\cmTC_57ba5.vcxproj]
```

After trying to uninstall everything pertaining to the project (Visual Studio, CMake, OpenCV files) I attempted to reinstall everything following the tutorial and try again. Unfortunately, this led to the same error. Eventually I copied the assignment over to another computer and tried continuing from there. Although this was a successful workaround, I was unable to fix the error on the original computer.

While implementing tracking into the program I faced several logical errors. The hardest to overcome was keeping the consistency of the trackers. I had issues where if an earlier tracker was lost, the others would be as well. This was because on a failed attempt to track, all proceeding trackers weren't drawn. When trying to fix this I created a new problem where if an earlier tracker was lost the colour from that tracker would be adopted by the later tracker. Eventually I created a class that held the objects attributes together (the colour, tracker, and bounding box). This allowed the class to keep the data organized, and prevented a loss of data when an object was no longer required.

HOW TO BUILD AND MAKE CHANGES

In order to make changes to the program, a programmer would first need to setup the environment to do so. To do so, there is a tutorial called "Install OpenCV on Windows" that I followed. Some of the steps can be passed over, as they are meant for Python projects. The software/files required in the tutorial includes Visual Studio 2015, CMake, the opencv and opencv_contrib libraries (3.4.4 was the version used in my case), and the files included in my FaceTracking.zip file.

Once the tutorial is complete the programmer should have been successful in compiling the example provided. If so, my program can simply be compiled (on Windows) using the included BAT file 'FirstRun.bat' after extracting the contents of the zip file into a folder. This will create a build directory, compile the program inside, then move the harr cascade file to the Release directory and run the program. To run the program again the user can run 'Run.bat', or run the new executable file located in the build/Release directory.

In order to make changes to the program the file FaceTracking.cpp can be edited. Assuming the FirstRun.bat has been run, changes can be tested by running 'CompileAndRun.bat'. It is recommended however, that if there are any errors to run the command 'cmake --build . --config Release' within the build directory using a PowerShell. This will allow time to read the errors.

RESOURCES

Crawford, Patrick W. “OpenCV for Python Developers.” *Lynda.com*, Lynda.com, 22 Sept. 2017, www.lynda.com/Python-tutorials/OpenCV-Python-Developers/601786-2.html.

Chandel, Vaibhaw Singh. “Install OpenCV 3 on Windows.” *Learn OpenCV*, 26 May 2017, www.learnopencv.com/install-opencv3-on-windows/.

Mallick, Satya. “Object Tracking Using OpenCV (C /Python).” *Learn OpenCV*, 13 Feb. 2017, www.learnopencv.com/object-tracking-using-opencv-cpp-python/.