

Portrait Machine

Robin Leverton

<https://vimeo.com/704741556>

A Raspberry Pi and Arduino Powered machine to Track Faces and Draw Portraits

Introduction

A machine to video onlookers, track their faces and display a line drawn interpretation of the image. Upon command it can then send gCode instructions via serial port to the Arduino powered pen plotter. The machine is interacted with using a touch screen.

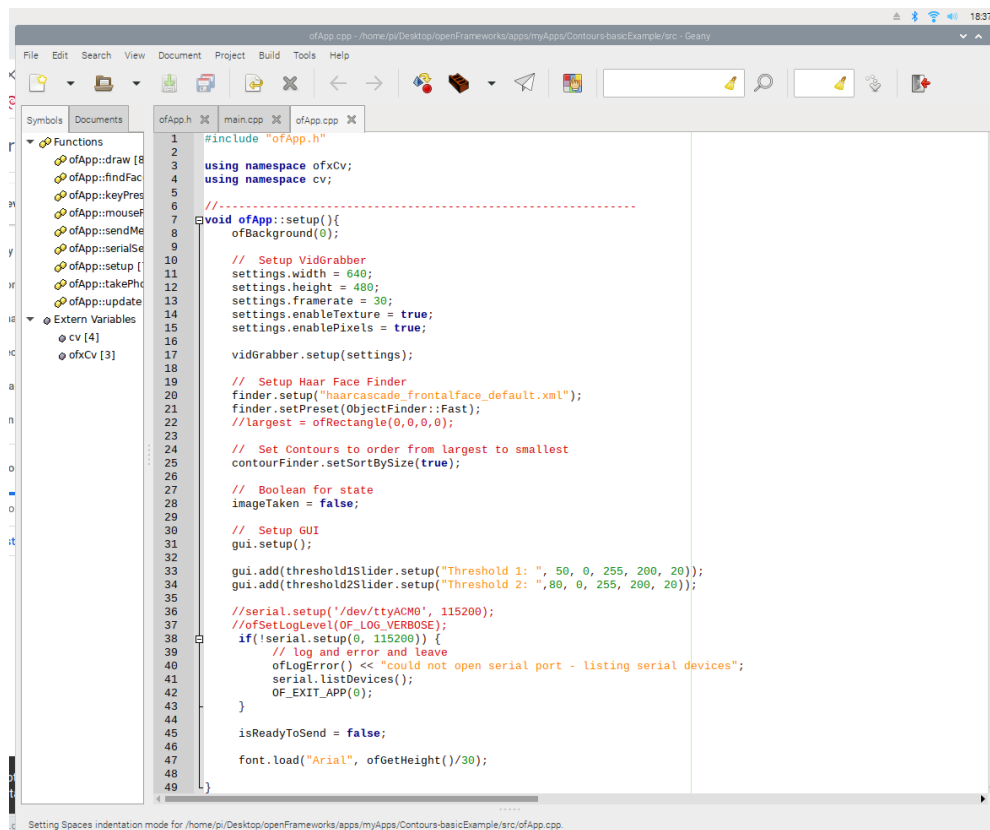
Concept

I had created a pen plotter for the Physical Computing 2 module which ran using Python. My idea for this project was to port the code to OpenFrameworks C++ and expand it to make a more selective output, that of taking and drawing portraits of its users.

Portraiture is a particularly personalised form of image making; There is a sympathetic connection between the portraitist and the sitter. The machine, singing as it works, carefully and confidently tracing out the contours of a sitters, face provides a captivating revelatory connection between the two. And its digital hand translates the computational eye of a camera into a medium with the direction, sensitivity and purpose of a pen drawing.

Technical Implementation

As this project was to run off a Raspberry Pi, the first step was to compile OpenFrameworks on the pi 3b+ that I have. Following the instructions proved easy enough, but the ofxOMXCamera addon for using the Raspberry Pi camera module required an earlier version of OpenFrameworks (V.0.10.* rather than the v.0.11.* that is the default installation) requiring a slight change in the bash script for downloading.



```
1 #include "ofApp.h"
2
3 using namespace ofxCv;
4 using namespace cv;
5
6 //-----
7 void ofApp::setup(){
8     ofBackground(0);
9
10    // Setup VidGrabber
11    settings.width = 640;
12    settings.height = 480;
13    settings.framerate = 30;
14    settings.enableTexture = true;
15    settings.enablePixels = true;
16
17    vidgrabber.setup(settings);
18
19    // Setup Haar Face Finder
20    finder.setup("haarcascade_frontalface_default.xml");
21    finder.setPreset(ObjectFinder::Fast);
22    //largest = ofRectangle(0,0,0);
23
24    // Set Contours to order from largest to smallest
25    contourFinder.setSortBySize(true);
26
27    // Boolean for state
28    imageTaken = false;
29
30    // Setup GUI
31    gui.setup();
32
33    gui.add(threshold1Slider.setup("Threshold 1: ", 50, 0, 255, 200, 20));
34    gui.add(threshold2Slider.setup("Threshold 2: ", 80, 0, 255, 200, 20));
35
36    //serial.setup("/dev/ttyACM0", 115200);
37    //ofSetLogLevel(OF_LOG_VERBOSE);
38    if(!serial.setup(0, 115200)) {
39        // log and error and leave
40        ofLogError() << "could not open serial port - listing serial devices";
41        serial.listDevices();
42        OF_EXIT_APP(0);
43    }
44
45    isReadyToSend = false;
46
47    font.load("Arial", ofGetHeight()/30);
48
49 }
```

Figure 1 Geany IDE

The ofxOpenCv lacks several OpenCV features I wished to exploit, such as Canny edge detection, thus utilising Kyle MacDonald's ofxCv was necessary in executing the project as envisioned.

My first hurdle was the communication with the camera module and then translation of that to a format with which OpenCv can work. The ofxOMXCamera addon basic demo example proved a success, however the ofxCv example failed to compile.

I was coming across an X11 issue, and my attempts to follow OF forum guidance didn't render any results.



Figure 2 Canny Edge detection test on laptop using ofxCv

The solution I found to the Linux ARM X11 problem turned out to be a matter of copying an ofxCV example folder "basic-contours-example" and then writing all my code within this. I don't know why this works when a blank project with simply the addons placed in the addons.make file and header will fail.

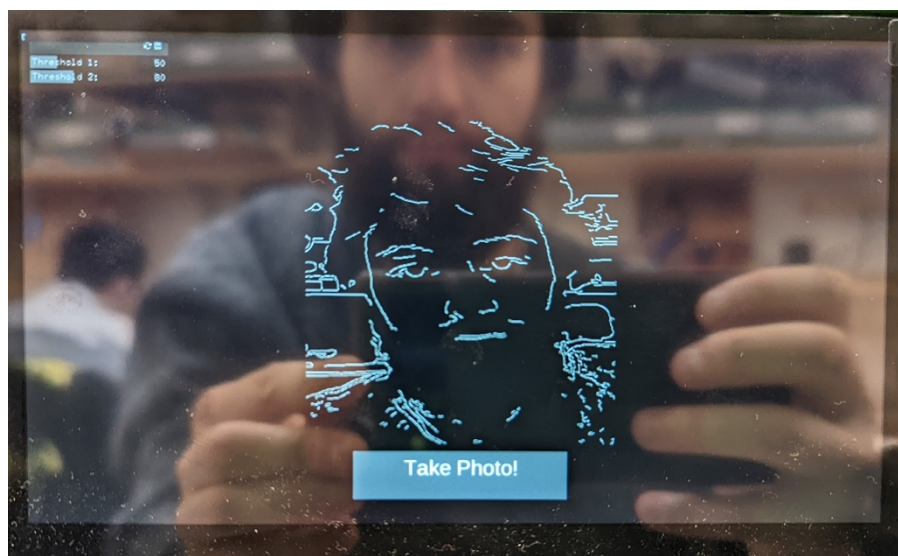


Figure 3 Photo of Contour-Drawn Image

With the camera input fed to ofxCV's face detection, I cropped the result to an area around the face before sending it through canny edge detection (See figure 2).

This output is then normalised and passed through cv contour finder. This provides me with an array of point arrays giving positions to be multiplied by the drawing area size. These contours and positions are looped over and output to a gCode file, which is then opened, read and sent line by line via serial to the Arduino based drawing machine.

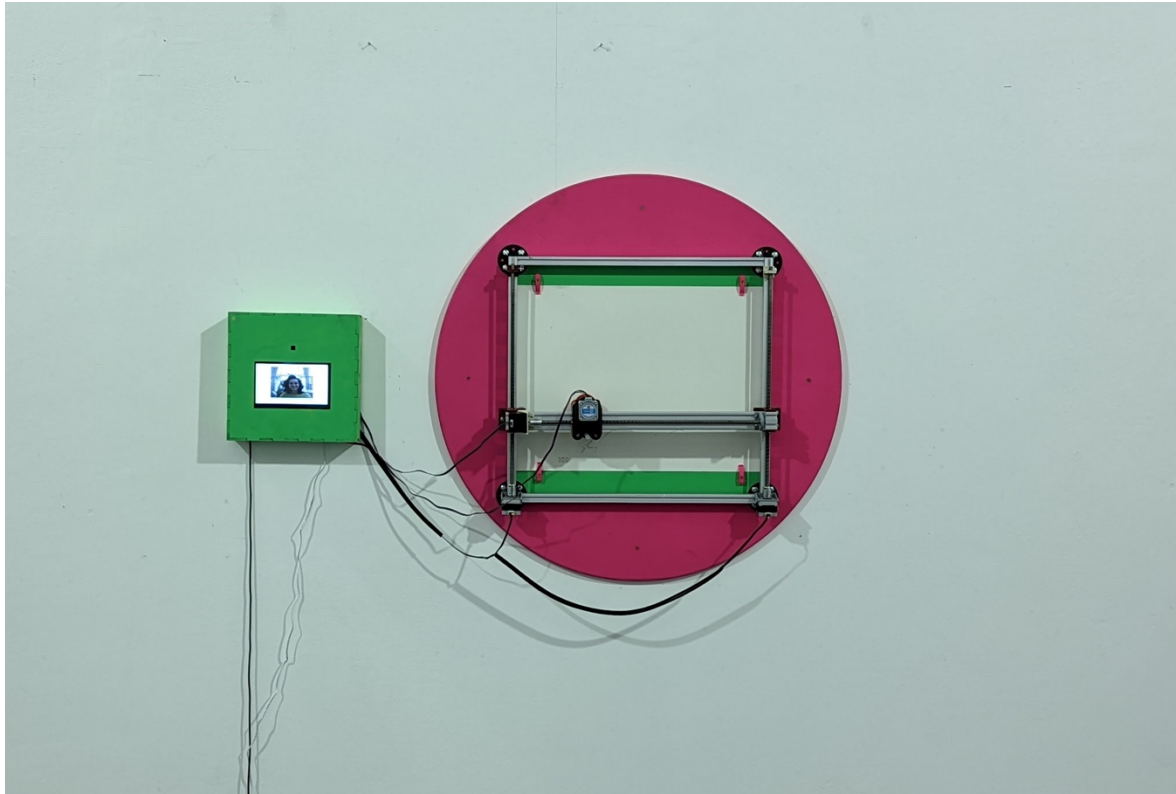


Figure 4 Drawing Machine Installation View

The serial communication was difficult to implement and borrowed a lot from the ofxGRBL2 addon written by Roy Macdonald. The addon itself doesn't work, so I took and adapted the sendMessage and update functions from its source .cpp file.

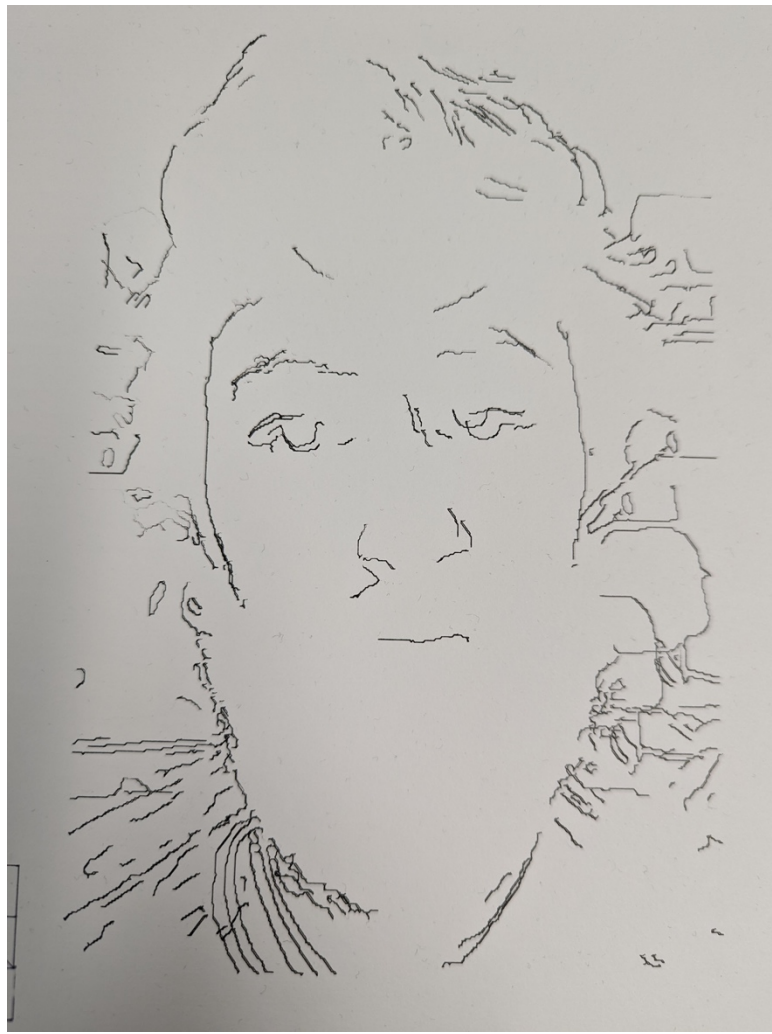


Figure 5 Finale Drawn Result

```

1370 //-----
1371 struct dirent *eps;
1372 // fallback to /dev/input/eventX since some vnc servers use uinput to handle mouse & keyboard
1373 typedef int (*filter_ptr)(const struct dirent *d);
1374 filter_ptr mouse_filters[2] = { filter_mouse, filter_event };
1375 string devicePathBuffers[2] = { /*"/dev/input/by-path/",*/ /*"/dev/input/" */ };
1376
1377 for(int i=0; i<2; i++){
1378     int n = scandir(devicePathBuffers[i].c_str(), &eps, mouse_filters[i], dummy_sort);
1379
1380     // make sure that we found an appropriate entry
1381     if(n >= 0 && eps != 0 && eps[0] != 0) {
1382         string devicePathBuffer;
1383         devicePathBuffer.append(devicePathBuffers[i]);
1384         devicePathBuffer.append(eps[0]->d_name);
1385         mouse_fd = open(devicePathBuffer.c_str(), O_RDONLY | O_NONBLOCK);
1386         ofLogNotice("ofAppEGLWindow") << "setupMouse(): mouse_fd=" << mouse_fd << " devicePath=" << devicePathBuffer;
1387         break;
1388     }
1389 }
1390
1391 if (mouse_fd >= 0) {
1392     char deviceNameBuffer[256] = "Unknown Device";
1393 }

```

Figure 6 Change to line 1376

An issue which arose when wishing to interact with the program through a touch screen was where Open Frameworks declares the mouse input to be used. I was able to find the solution on the OF forums that suggested altering a couple of lines in the “setupNativeMouse()” and “updateNativeInput()” function (Figure 6 / 7).

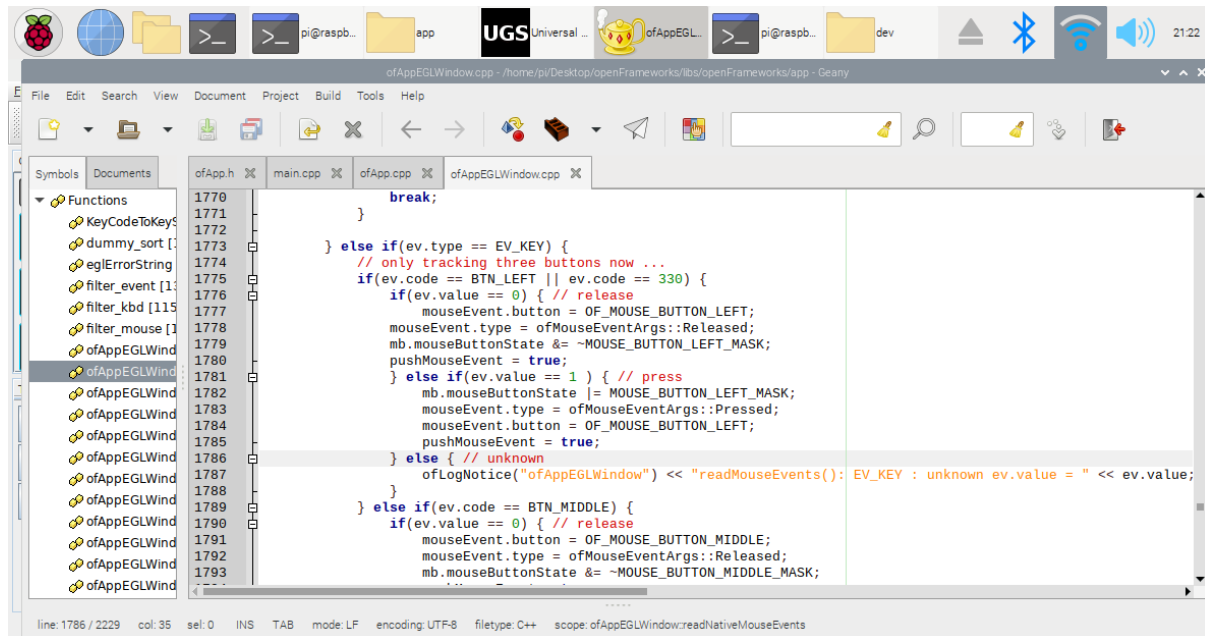


Figure 1 Line change to 1775 addition of `ev.code==330` to left mouse button check

Reflection

The project was successful. Porting the spine of the program across to C++ was much more difficult than imagined and so was the aspect of writing and compiling everything on the Raspberry Pi.

I had thought it would be a matter of finding the comparative OpenCv functions from ofxOpenCv, however, the wrapper of ofxCv proved to be much more usable once I had gotten around my issues of not compiling. I had also thought the camera module of the Raspberry Pi would simply act the same as a webcam and be accessible by the ofVideoGrabber function, so having to download a specific addon and deal with its complications was a surprise.

I was happy with how my knowledge of image types and the ofImage and Mat classes grew when altering the photos and passing them through different functions. The obstacle provided by ofSerial communications was made clear by the cannibalising of the ofxGRBL2 addon, which in itself wouldn't work but provided the framework to explain how to send and read bytes over the serial connection.

For future development, I would be looking to make a smoother interactive experience, perhaps by using voice commands to tell the machine when to take and if to draw the photos.

References

Fisgus, Felix, and Joris Wegner. 2022. "EXPLORATIONS IN AUTOMATED ARTISTIC PORTRAIT DRAWING". *Felixfisgus.De*.
https://felixfisgus.de/user/downloads/Masterthesis_JW_FF.pdf.

MacDonald, Kyle. 2020. "Github - KylemcDonald/Ofxcv: Alternative Approach To Interfacing With Opencv From Openframeworks.". *Github*.
<https://github.com/kylemcDonald/ofxCv>.

MacDonald, Roy. 2020. "Github - RoymacDonald/Ofxgrbl2: Openframeworks Addon For Easily Sending Gcode Commands To A Device With GRBL Firmware". *Github*.
<https://github.com/roymacDonald/ofxGrbl2>.

"Ofxcv Macro Error On X11". 2017. *Openframeworks*.
<https://forum.openframeworks.cc/t/ofxcv-macro-error-on-x11/26022/16>.

"Touchscreen On A Raspberry Pi". 2015. *Openframeworks*.
<https://forum.openframeworks.cc/t/touchscreen-on-a-raspberry-pi/19827/14>.