|  |  |  |
| --- | --- | --- |
|  | Android App | C++ Desktop App |
| Development | * Created in Android Studio. * Written in native Java/XML * Uses opencv-3.4.10-android-sdk | * Created in Visual Studio * Written in C++ * Uses opencv-4.2.0 |
| Performance | * Sluggish compared to PC * openCV for Android (Java) a lot more “finicky” compared to C++ libraries * In Android, you get a continuous feed from the device’s camera and have to intercept each frame to run algorithms on to find contours and print lines identifiying each contour | * Performance much better compared to Android due to more processing power & ram * in C++, you start a VideoCapture module and snap/print frames. This is smoother. |
| Concepts used | * Edge detection * Binary thresholding * Blurring | * Edge detection * Binary thresholding * Blurring |

**Mobile App** *vs* **C++ Desktop App**

one of the methods we tried was to make an Android app using Java. While the app didn’t work out so well in terms of performance, it provided a way to make the recognition program available on portable devices where it can get a vantage point in usability.

One of the major reasons why the android app didn’t work was because of the openCV libraries we had to use. The C++ libraries are much more efficient as openCV itself is written in C++. the libraries for Java are a lot more finicky and require a lot of circling around to make even the most intuitive of tasks work. Furthermore, the android app is designed to get a continuous feed from the device’s camera. Meaning you have to intercept each individual frame to perform algorithms and print the idenfitication lines.