CS CAPSTONE TECHNOLOGY REVIEW

NOVEMBER 21, 2017

CDK DATA STREAM AI

PREPARED FOR

CDK GLOBAL

CHRIS SMITH

PREPARED BY

GROUP 65

SIGFIND

JACOB GEDDINGS

CONTENTS

1 LANGUAGE OF OPENCV

1.1 Overview

This portion of the document will go into a technical review of three potential options for OpenCV supporting languages. The options are C++, Python, and Java; the pros and cons of each will be noted. Once all options are considered, there will be a discussion that directly compares these three choices. Lastly, a conclusion will be drawn from these three choices regarding the leader that we will utilize going forward.

1.2 Criteria

Our project has a hard requirement that the AI be open sourced. As such, we are restricted to utilizing established AI platforms. With this in mind, OpenCV is the leading candidate primarily due to its flexibility and popularity when wanting to handle image processing. When factoring in that OpenCV is our AI of choice, there are only three language options that are viable: C++, Python, and Java. Thus at present our only feasible options for supporting the AI are to choose between these three programming languages.

1.3 Options

1.3.1 C++

C++ is the primary language used by OpenCV and the most robust in general. C++ is widely regarded as one of the most efficient languages when it comes to runtime resources. It also carries the largest selection of tutorials and guides to use for picking up and using OpenCV. The library supported under C++ is massive when compared to most other languages. Lastly, the community backing it is extensive. This option has been available for years, and thanks to its status as free-to-use and its efficiency, many users have taken to working with it. As a result, many questions have been answered already regarding the usage of OpenCV within a C++ language. Some issues do arise with the C++ variant with the largest being poor documentation. Another complaint is that while it supports a strong library, its actual pool of machine learning components is rather limited by comparison. This results in difficulties when attempting to use multiple different learning routines on a program. Another issue is the challenge of interpreting code and the debugging process. While this is a normal issue for any programming language, for C++ it is compounded when attempting to make use of machine learning algorithms. If the programmer needs to write any code from scratch, this method becomes a nightmare for comprehension.

1.3.2 Python

Python is the leading rival of C++ in language preferences. The most popular strength of Python is the simple ease of use associated with it. The language is straightforward and easy to write in. Another interesting fact about Python is that its often regarded as a language of scientific computing, with support for packages such as OpenCV, numpy, scipy, scikit-learn, and matplotlib[1]. In direct contrast to C++, the debugging and visualization of Python is considered very good which is primarily the result of its previously mentioned simple-to-understand syntax. Also, portability is

something that should be considered with our program, and as such popularity and use of Python means that most systems already have support for this language. However, the documentation is lacking even more than the C++ option. There are concerns including difficulty discerning what a function does on a deeper level, what the various parameters do to impact the program, and relatively weak guides in the usage of this version.

1.3.3 Java

Java is the final language being considered for use within OpenCV. This language as a choice is a bit different from others being considered in that, while all lack documentation, this one takes it to the furthest extreme. Having not existed until version 2.4.4 of OpenCV, Javas introduction is the latest of the bunch. Its primary focus is on its usage in Android devices[2]. When attempting to research this language in particular, there was very little mention of it at all, with tutorials and guides being few and far between. While Java itself is a reliable enough language, it has failed to gain traction when placed in an OpenCV environment primarily due to its middle-of-the-ground nature. What this means is that its runtime performance is generally in-between C++ and Python. This means if performance was a concern, then C++ is the better option. When considering ease of coding, then Python wins, with Java falling into second place for either category. This is only made worse by the previously mentioned lack of documentation, which means it is extremely difficult to make proper use of this language. While other languages also suffer from this, they at least have several years of exposure in which the documentation has been improved or added. Java on the other hand has had minimal additions.

1.4 Discussion

These three choices are largely different from one another with the only considerable common ground being that they all have poor documentation in general. The discrepancy however, is that of the three options C++ does carry the strongest online support and Java is at the bottom by a significant margin. Next for consideration is ease of use for coding purposes. In this aspect, Python is the strongest candidate and Java and C++ are the ones trailing behind. When considering code efficiency at runtime, then C++ is the clear winner.

1.5 Conclusion

Given what is known about the three choices at present, the current leader that will be selected is C++. There is also the possibility that since Python utilizes C++ under the hood, should we need to try and move to a different language due to library restrictions the migration wouldnt be too taxing as opposed to involving C++ or going from Python to C++.

2 Filter Algorithm

2.1 Overview

This section will be focused on the consideration of image blurring techniques in OpenCV to assist in dealing with potential noise that may exist on the documents well be scanning. To combat this issue and make the image more

readable for the program, well need to make use of some form of blurring technique in which there are three major choices.

2.2 Criteria

Whatever technique is being utilized must be functioning on OpenCV. The blur in question must also be safe enough as to not disrupt the natural look or flow of the document, should it become too extreme text bodies will appear completely connected. Lastly, the technique used must be efficient enough as to not impact overall performance to a point where the program cannot keep up with potential inputs.

2.3 Options

2.3.1 Gaussian

Gaussian is one of the most common implementation methods for blurring an image. This method takes a defined kernel and slides it across the image to smooth it out [3]. This kernel needs to be defined by the programmer with the parameters sigmaX and sigmaY. These values define their respective x and y scale. In addition, should the parameter for y be left undefined, then it will inherit the value of x creating a square shape [4].

2.3.2 Median

Median filtering is another common selection for its ability to carry out limited edge detection to help preserve the form of a given shape. While not perfect at this task it is often sufficient at preserving shape while also removing noise [5]. This method also carries a relatively new function built in to OpenCV that processes all channels of the image input independently [3]. This process is best at tackling the issue of images with heavy salt-and-pepper noise in which the central median value is always replaced by some pixel image, unlike most other filter methods [4].

2.3.3 Bilateral

Bilateral filtering is very effective at removing noise within an image while still retaining the edges to everything. This comes at the price of significantly slower runtime to complete when compared with competitors. The procedure is done through the use of two gaussian filters, one for space to handle nearby pixels and the other for intensity differences. This allows it to blur everything but areas with high differences which naturally indicate an edge in the image [4].

2.4 Discussion

When comparing these three filter choices the areas that need to be considered are runtime performance and the importance of keeping the edge to an image. If the only concern is performance then Gaussian would be the clear choice, but since were dealing with the scanning of text documents, the situation becomes a bit more challenging. At present my assumption is that to correctly detect potential signatures, well need to be able to properly see the relative shape of words which means that the edge will be vital to accomplishing this task.

2.5 Conclusion

Since our greatest concern is determining if it is even possible to accurately detect if a page has a signature section and if its been properly signed then performance will be taking a back seat to simply completing the task. As such, our best bet is to use bilateral given it effectively removes noise while still maintaining the form of the document to its most original state.

3 File Conversion Software

3.1 Overview

CDK stores all submitted documents as PDF formats which is not a readable file type with OpenCV. As a result, we will need to consider ways of converting the PDF into something compatible. Several resources already exist in this department that show potential use for our purposes, but determining which to select is the current problem.

3.2 Criteria

It is established that the documents well be dealing with are of the PDF format. Beyond this, the team is free to select however we see fit to implement this portion. Our current preferences indicate the use of C++ and OpenCV so well need to choose based on what best conforms to the language and platform.

3.3 Options

3.3.1 ImageMagick

ImageMagick is a free open source option for image conversion and editing. It carries the functionality to read and write over 200 formats including PDF, JPEG, and PNG. The software carries command line support and works with all three considered language options that our team is considering. It also functions on Linux, Windows, OSX, iOS, and Android devices [6]. With all of this functionality comes a major drawback however: it is a very heavy program to use. While it would be nice to have all these options available to us, we only plan a single PDF to image conversion and that would be wasting the majority of what makes this program so taxing to operate.

3.3.2 Ghostscript

Ghostscript is a copyrighted software package that can convert PDF files as well. It can function as an interpreter for PDFs and turn them into PostScript files [7]. To utilize this PostScript, we would then need to search for an addition software package for converting it into a suitable format for OpenCV. As such, while a lightweight program on its own, it will require a daisy chain of conversions to make it ready for manipulation on our end.

3.3.3 MuPDF

MuPDF is an open source software package that is written in C with the intent of doing PDF conversions. This program is an extremely lightweight PDF viewer and editor with the ability to convert the file into various formats such as HTML, SVG, and CBZ. In addition to C, there is support for Java and Android functionality with this program [8]. Unfortunately, to complete the conversion there will need to be an additional conversion of SVG into a format that is accepted by OpenCV.

3.4 Discussion

Out of the three options provided one of them is not open source which is cause for concern given the project aims to be an open source product. As such, to be safe in our decisions the choice falls between MuPDF and ImageMagick. ImageMagick is the all-in-one package with the support for what we need (and then some), whereas MuPDF would only get us closer to the proper image format that is required. While ImageMagick achieves our goal immediately, its added features make it a significantly more taxing program to use, a factor that we must consider.

3.5 Conclusion

While efficiency is a large concern for us given the scale of the project and how computationally heavy artificial intelligence is, we do need to consider that we primarily just want to prove this project can be done. As a result, ImageMagick is the software we will be going with since it stays safe in open source and reduces the amount of work required for us to chain several software together to get an input we can actively use within OpenCV.

4 References

[1] ”LearnOpenCV” Internet: https://www.learnopencv.com, Oct. 30, 2015 [Nov. 14, 2017].

[2] ”Introduction to Java Development” Internet: https://docs.opencv.org, Nov. 13, 2017 [Nov. 14, 2017].

[3] ”OpenCV Tutorial C++” Internet: https://opencv-srf.blogspot.com, [Nov. 21, 2017].

[4] ”Smoothing Images” Internet: https://docs.opencv.org, [Nov. 21, 2017].

[5] ”Filters B” Internet: http://www.bogotobogo.com, 2016 [Nov. 20, 2017].

[6] ”Image Magick” Internet: https://www.imagemagick.org, 2017 [Nov. 21, 2017].

[7] ”Ghostscript” Internet: https://www.ghostscript.com, 2016 [Nov. 20, 2017].

[8] ”MuPDF” Internet: https://mupdf.com/, 2017 [Nov. 21, 2017].