Oregon State University
College of Engineering

CS Capstone Design Document

December 1, 2017

CDK Data Stream AI

Prepared for

CDK Global

Chris Smith

Prepared by

Group 65

SigFind

Inhyuk Lee

Jacob Geddings

Juan Mugica

Abstract

The purpose of this design document is to outline the various pieces weve previously covered in our technology reviews and add a projection of what our plans are regarding the decided upon technology for each piece. Given our research centric project, we will be taking care to mention potential alternatives should a decided upon technology prove ineffective at accomplishing our goals. Each piece of technology discussed will be done so under a general design viewpoint within which it fits.

CONTENTS

1 OVERVIEW

1.1 Scope

This document will cover nine pieces of the project including our chosen technology to accomplish that task and projected plans regarding that technology. This will include plans for if the technology fails to be what we anticipated as well as plans of dropping the piece down the line should it no longer be relevant to our needs. These pieces will each be discussed from a defined perspective or viewpoint that best defines the technology in question. Some examples of viewpoints include a dependency viewpoint, algorithmic viewpoint, and composition viewpoint.

1.2 Purpose and Background

Planning is vital to the success of this project. Each component should be considered and have backups in place should our predictions go awry. As such, this design document helps illustrate our projected plans and security precautions. This will help give clarity for our client as well as ourselves in seeing how we intend to progress and gives us clear markers for when we have achieved a given task. While this document will not exhaustively cover every edge case, it does touch upon nine vital areas our group has been concerned about for completing the project.

1.3 Intended Audience

The primary audience for this document will be our instructors and client who have moderate to high levels of expertise within this field. This document will help in giving clarity regarding our intended actions in the following months as we enter the coding and development phase of the project. In addition, this document will operate as a checklist within our own group to ensure we are staying consistent with our documented plans and with what we actually produced.

1.4 Glossary

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

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

[3] ”CDK Global” Internet: https://www.cdkglobal.com/, 2017 [Nov. 10, 2017].

[4] ”Python” Internet: https://www.python.org/, Oct. 17, 2017 [Nov. 12, 2017].

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

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

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

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

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

[10] "MuPDF" Internet: https://mupdf.com/, 2017 [Nov. 21, 2017].

[11] "OCR Optical Character Recognition Internet: http://www.recogniform.net/eng/ocr-optical-character-recognition.html, 2016 [Dec. 1, 2017].

[12] "System software: User interfaces" Internet: https://en.wikibooks.org/wiki/A-levelComputing/CIE/Computersystems,commu 2017 [Dec. 1, 2017].

2 DESIGN VIEWPOINTS

2.1 Introduction

The viewpoints model of depicting our various pieces for this project gives the opportunity to plan for that piece in regards to what role it must serve. The various viewpoints depicted below show what the intended function for that technology is for the group and how it will be used to serve the project as a whole. Each viewpoint carries its own distinct subset of rules as well to help clarify the particular function it needs to carry out.

2.2 Context Viewpoints

2.2.1 Design Concerns

Each piece of technology is outlined in its purpose as well as its potential shortcomings. Depending on the viewpoint being utilized for that given piece, the exact specifications of this section vary, but the core of it remains the same. We present the problem, the potential restrictions that exist, and our intended technology to remedy this while not violating the restrictions in place. In addition, it will include alternatives should our predictions be wrong.

2.2.2 Design Elements

In this section of a provided viewpoint the technology in question will be described in greater detail. Indications of what strengths or weaknesses an item may have are described here as well as the alternative technologies abilities. Beyond this portion, depending on the viewpoint being used there are additional potential subsections that may follow.

2.3 Languages of OpenCV(Composition)

2.3.1 Design Concerns

The largest concern in this design is the fact that OpenCV is primarily restricted to only a few languages. C++, Python, and Java are the three candidates worth considering, and there are significant differences between them. When choosing the language for our project, we have to consider two major factors: AI is a computationally heavy process, and we need to primarily prove that the task can be done (not necessarily in the most efficient manner). After consideration of these factors, we decided to go with C++ as our primary focus given its documentation and optimization [1], but a backup plan has been made for Python migration.

2.3.2 Design Elements

Design entities:

The primary entity is the language itself, C++, with its supported libraries as an extension of that. C++ is the leading language in extensive library support which helped lead to its design choice. Python is the backup here with a weaker overall support, but it maintains a competent package for AI usage [2]. In either case the environment for working on the project will not change; both function in the same space and will not require special equipment to complete the coding.

Design relationships:

The language will define how the entire project proceeds. It will be involved in all components of our project. Peripherals to this project such as Docker support and cloud storage are considered low priority; CDK can easily handle those portions on their own [3]. The program will be designed with modularity in mind, however should this program be expanded upon, we will need to reduce the difficulties involved with accomplishing that task.

Design attributes:

C++ operates in a predictable, expected manner. Maintaining an up-to-date grasp on its functionality is relatively difficult due to its memory allocation and syntax requirements, but it does result in some of the best performances at runtime. Python (as a backup) does carry some differences from what might be traditionally expected. For OpenCV to function properly on Python, it will utilize C++ significantly just out of view of the programmer through the use of library packages like numpy [4].

2.3.3 Data Attributes

The internal functions of C++ are very flexible by nature despite the syntactic difficulties of writing in it. A fair bit of this program will be dynamic in nature; the AI will be learning from a constantly changing database of images and as such must be flexible in its capacity to interpret the data. There will be some static elements to this program as well, including image conversion and image filtering. All received files will be in PDF format and will be text documents that will need to be scanned. This allows for the aforementioned components to remain static.

In the event Python is incorporated into the project, presumably due to complications with C++, the attributes will be mostly the same as well. The process will be identical, but the functions used will have to be different. If this direction is taken with the project there will be a notable slowdown in performance regarding all aspects of our project due to the nature of Python.

2.4 Image Filtering(Algorithm)

2.4.1 Design Concerns

The leading concern for this project is whether the task assigned to us can be completed. As such, the focus for image filtering is placed on how effective it is at assisting the program in finding the signature windows. It is not a primary concern at present to have the most optimal algorithm used for the smoothing of our images, just one that will be most likely to succeed. Should we be incorrect in our assumptions for what would work best, there are another two functions that could be used as substitute. Should the project be proven possible, consideration will be made for using more efficient functions without losing accuracy on program readouts. Median will be the second choice due to the fact that it carries slightly stronger properties at shape retention through the blurring process. If both preceding options fail then Gaussian is unlikely to improve the situation but will be attempted regardless [5].

2.4.2 Design Elements

For this portion of the project we are interested in improving readability for the AI in the event the submitted images are filled with too much noise and it cannot correctly interpret what is being fed to it. To accomplish this, there exist several different functions that seek to smooth out a provided image at varying computational costs. The leading choice for our needs is bilateral filtering, a method that retains edges very well while blurring everything else. Fallback options in the event of failure include median and Gaussian which are cheaper to run on a system but provide a significantly stronger blur that can disrupt the original shape of an object within the image. Median still makes an effort to preserve object form while removing static-like noise in an image, but it doesnt make shape retention a high priority. Gaussian by itself is a very harsh method of blurring for our purposes in that it does not care about original form of the image; it effectively passes a thick brush across the image and that is it [6].

2.4.3 Processing Attribute

Bilateral filtering can effectively remove noise whilst retaining the object within the image. It is a costly function call however; bilateral filtering functions via calling two unique gaussian filters and designating one to sweep through the space of the image and search for sharp changes in pixel intensity which indicates the edge of an object. The other filter will move through the image and blur whatever does not show a steep change in pixel intensity. For this process to work in a reasonable manner, it will be implemented with the same language as the AI and be handed the image conversion of our PDFs. The output should enable the AI program to more readily see the distinct objects on the image and, most importantly, determine the locations of signature windows. In the event that bilateral does not accomplish the task, the use of a median filter is our next best option. Median functions under a relatively new method for OpenCV in which it processes all channels of the image input independently. This process as a result can frequently blur any salt-and-pepper like noise on an image without breaking up object form [7].

2.5 Computer Vision Library(Composition)

2.5.1 Design Concerns

Choosing computer vision library is critical for designing signature detecting software. Using a robust library will improve the functionality of the software and reduce the development time. To select good quality of library, three common computer vision libraries are compared. These libraries, TensorFlow, Deep Learning 4 Java(DL4J), and OpenCV are evaluated based on: flexibility of environment and language, variety of functions, weight of the functions, and the size of the community. When considering these factors, OpenCV library was chosen for this project as the strongest candidate.

2.5.2 Design Elements

Design entities:
OpenCV is an Open Source Computer Vision Library that provides various functions related to digital imagery and video. Since the main goal of this project is recognizing signatures on a document, this library will be vital in correctly analyzing any submitted form. OpenCV supports many languages such as C++, C, Python and Java but it is optimized primarily for C and C++. Also, it is executable on both desktop (Windows, Linux, Android, MacOS, FreeBSD, OpenBSD) and mobile (Android, Maemo, iOS) [10].

Design relationships:
Computer vision library have high relationship with design and efficiency of the algorithm. Each computer vision library supports different functions to solve the same problem. Based on what library that software is using, design of the algorithm will change. In addition, since each library supports different functions, the performance of each functions varies. Therefore, choosing a good library will increase the performance of the entire software. The interaction between environment and library is also important. Since different libraries supports different languages and operating systems, our choice will significantly impact our options when developing the project. OpenCV has high flexibility on environment. It supports four languages and runnable on nine operating systems.

2.5.3 Design Attributes

Unlike other computer vision libraries, OpenCV been functioning for years with continued support. Because of this, OpenCV carries with it a large and diverse community of supporters. According to the OpenCV website, there are 47 thousand people in the user community, and more than 14 million downloads [10]. Since there is such a large community, learning OpenCV and fixing issues will likely be easier than with less documented or supported formats. OpenCV is very fast at runtime when compared to other AI platforms. This is because OpenCV library is written in C and C++. It also supports 500 improved algorithms to accelerate the original algorithms [10]. OpenCV do this by using CUDA which is modern GPU accelerators that increase GPU performance, so an algorithm can be computed in a short period of time.

2.6 Text Recognition Method(Algorithm)

2.6.1 Design Concerns

Choosing computer vision library is critical for designing signature detecting software. Using a robust library will improve the functionality of the software and reduce the development time. To select good quality of library, three common computer vision libraries are compared. These libraries, TensorFlow, Deep Learning 4 Java(DL4J), and OpenCV are evaluated based on: flexibility of environment and language, variety of functions, weight of the functions, and the size of the community. When considering these factors, OpenCV library was chosen for this project as the strongest candidate.

2.6.2 Design Elements

Design entities:
Intelligent Character Recognition(ICR) is a text recognition algorithm. It converts text images to electronical text documents by taking three steps. It first analyzes the image, recognizes the paragraphs, text lines, and words from the image. Then, it splits the images into individual characters. On the last step, it compares image characters with characters in the dictionary, and converts the characters [11].

Design relationships:
ICR directly impacts the performance and speed of the software. Since ICR is the algorithm that is used for detecting the signature box, the higher accuracy of the character recognition will lead to the higher accuracy for detecting the signature box. It also impacts the speed of the software. If the algorithm is heavy, speed of the software will go down. ICR algorithm does have comparatively high accuracy and high-speed when looking at its competitors.

2.6.3 Design Attributes

For detecting characters, ICR has three features to increase the accuracy of the function. First it compares the image character with character in dictionary. Second, it evaluates the characteristics of the character such as number of lines and angles between lines to determine which character is it. Last feature is the self-learning feature. ICR learns through its experience to increase accuracy. Also, since it can learn different font and style, it can determine handwritten texts from the image.

2.7 User Interface Type(Logical Viewpoint)

2.7.1 Design Concerns

Selecting proper user interface is important to draw full functionality of the signature detection software by user. To choose the best user interface for the software, characteristics of the user and characteristics of the software were considered. Determining who is the user and characteristics of them is the one of the most important factor. Based on

their knowledge and experience, efficiency of the user interface will change dramatically. Also, software characteristics such as input constraints and hardware constraints also influence user interface. Based on the two characteristics, menu driven interface has been chosen among command line and form based interface.

2.7.2 Design Elements

Design entities:
Menu Driven interface interacts with user by providing simple selectable menus to user. It will show a small number of selectable menus to the user, and when user select the menu, it will perform the task that was written on the menu or gives the other selectable menus [12]. Menu driven interface is similar as tree structure. First menus act as a root of the tree, and on each menu, they may or may not have branch menus.

Design relationships:
As it mentioned on design concerns section, selecting proper user interface will increase the chance of user to perform pull functionality of the software, and considering characteristics of user and software will make higher chance for user to do this. Potential user of the signature detection software is employee of the car dealers. They have little or no knowledge on artificial intelligence or image recognition process. Therefore, user interface must be simple to use without background knowledge of image recognition. Signature detection software need to take the image document and give result of the signature detection, so user interface does not need to provide unnecessary ability to user. This will only draw confusion to user, and it might lead to malicious code.

Design constraints:
Since menu driven interface limits the users action, it also constraints the functionality of the software. However, signature recognition software has simple functionality, so limiting software control will not affect the efficiency of this software.

2.7.3 Design Attributes

The structure of the menu driven interface is simple and intuitive. User simply have to click on the menu that is provided, so it doesnt require any background knowledge. In addition, this interface constraints the ability of the user by giving limited option to user. This will reduce the confusion of the user and avoid malicious code.

2.8 Image Converting (Dependency)

2.8.1 Design Concerns

The program will need to begin operations using PDF files, which are not a valid file format when operating with OpenCV. To resolve this issue, the use of an external image conversion program will be necessary. Functionality is the priority in design and consideration for efficiency will be made after we can confirm the project is doable.

2.8.2 Design Elements

The leading candidate for image conversion is ImageMagick, which operates as a standalone image converter within C++ that is open source. With command line support, the program can convert our PDFs into JPEG which OpenCV can then correctly interpret [8]. It does require more computer resources to function compared to some competitors such as GhostScript and MuPDF[9][10]. However, it is important to note that ImageMagick will do the entire conversion in one go unlike MuPDF [10], and it is free to use unlike GhostScript[9]. As it stands, the existence of an image converter is necessary for us when handling the various example inputs provided to us, but the longevity of this portion in our project is questionable. It is possible that our client will want to incorporate their own image conversion process so plans to make this section of our project easily removable have been made.

2.8.3 Dependency attributes

In the final stages of this project, the dependencies on file conversion may go away when handing the AI program off to our client. However, it is vital until we reach that point. Without this software, we are unable to correctly build a library from which our project can read to make its assessments. As previously mentioned as well, OpenCV does not read PDFs in any capacity, and so to correctly analyze a document it must first be provided in a legible format for the program.

2.9 Runtime environment and Neural Net (Algorithm)

2.9.1 Design Concerns

The project being designed must be hosted in a runtime environment that is continuous as well as reliable. Given that we are taking an unsupervised learning approach to complete our task, it is imperative that it be able to run concurrently with instances of itself. The program must be designed in such a way that the ideal environment can host it and run it for extended periods of time. When taking this design paradigm into account, the algorithm running on this environment presents itself as the biggest design constraint. It must be optimized to be as noninvasive to the runtime environment as possible, while also efficiently making progress towards the desired outcome.

2.9.2 Design Elements

Design entities:
The first thing to take into account is what specific runtime environments can be chosen to perform this task. Careful consideration has landed our group to opt for OSUs flip server as opposed to other paid servers or personal machines. The second design entity is the algorithm itself. For this specific project we will be implementing a Convolutional Neural Network algorithm specifically suited for the processing of images, and qualities within them. Design wise, the neural net must be instantiated to run continuously, but in such a manner that does not exhaust our student resources on the OSU flip server. (e.g. utilize our allotted amount of memory) This will prove to be a key challenge since CNNs are extremely memory intensive algorithms, often handled by Graphical Processing Units as opposed to normal Computer

Processing architectures.

Design relationships:
How often we can deploy our algorithm, as well as how efficiently we can allow it to run on the given environment, presents itself as a critical constraint in regards to the final outcome of our efforts. If either of these aspects are handled sub-optimally it will inevitably affect how well our algorithm learns, as well as its performance as an end product.

Design attributes:
OSUs flip server is monitored 24/7 as well as available 24/7. It has high memory constraints for students and excellent support if there were to be unexpected problems. Convolutional Neural Nets are excellent at discerning patterns within pixel image data. The various layers can be dissected as levels of depth regarding the images attributes. It requires high memory usage

2.9.3 Data Attributes

Data Attributes:
The data being fed into the algorithm is of numerical 3-dimensional nature. This is essentially what an image is when processed in its raw form. The convolutional network that will be instantiated will handle numbers ranging from 1-255 per channel, (RGB), per pixel. The handling and processing of all these different ranges is the culprit for convolutional networks extremely heavy memory load. Our design must combine the processing of these numbers with the right amount of hidden neural layers so that the network functions properly, but does not overload the system.

3 CONCLUSION

For each piece of the project mentioned, the problem has been defined and a technology to remedy that problem has been identified. In addition, alternatives have been listed in the event we misjudged a given technology. Also, thanks to the viewpoints model being used, we now have greater clarity on how these various problems play in the bigger picture of the project and how best to approach them. While there are likely more parts to this project of which we are currently unaware, this provides a strong enough foundation to move forward into the development and coding phase of the project.