Computer Science and Engineering 2018, 8(1): 1-6
DOI: 10.5923/j.computer.20180801.01

Cross-platform Mobile Document Scanner

Amit Kiswani

Lead/Architect Mobile Applications, Paramount Software Solutions, Atlanta, USA

Abstract Document scanning is becoming more and more important these days. Most of the businesses need one or the other document like your identity, or some sort of proof (residence, income, SSN or birth certificate). Mobile phones, tablets, and iPads are easily accessible than the scanner connected to the personal computer, so it is crucial to integrate the scanning feature in the mobile applications. This paper briefly compares two possible approaches for augmented paper document scanning, Native versus Cross-platform and proposes a cost-effective solution, cross-platform mobile scanner, using technologies like Apache Cordova, OpenCV, HTML5, JavaScript, and CSS. The cross-platform solution provided for document scanning is possible to extend for a mobile website as well. This paper also presents different computer vision algorithms, fundamentals of image formation, camera-imaging geometry, feature detection and matching of digital images, and it also explains methods for acquiring, processing, analyzing and understating digital images.

Keywords Cross-platform, Native, Hybrid, Augmented, OpenCV, Computer Vision, Adaptive thresholding, Canny Edge Detention, Apache Cordova

1. Introduction

Mobile App market is tremendously growing. Most of the organizations and businesses have dived in adopting Mobile applications as the best mean of communication with their users. According to PEW research, 77% of U.S. Adults own a smartphone. This is the substantial motivating factor for organizations to incorporate all the best possible features in the mobile application to suffice their business needs. One of the crucial features that need to be assimilated in the mobile applications is the document scanning and submission. Many business industries like mortgage, insurance, cell phone, banks, home construction companies, healthcare, pharmaceutical and government agencies significantly require their users to submit various documents for different business reasons.

With smartphones comes camera, which gives a user an ability to take pictures. Adding the Ability to scan essential documents with mobile and upload there and then is a great and convenient feature for users. The dark side of it is that the different architecture and programing language of the software and the hardware capability of numerous smartphone devices pose a challenge for document scanning.

There are two different development methods to build the mobile application- Native and Cross-Platform. Native is considered to be the traditional method, but technology evolves rapidly leading to the rise of the new method, Cross-Platform. It is a crucial decision to determine the best method of development especially for the development of a scanning feature. Many companies are adopting Cross-Platform development framework for mobile apps, but there is no straightforward solution available for scanning feature in the Cross-platform framework. This paper will explain the need to choose the cross-platform method for scanning development and the elaborative steps to accomplish the scanning results via this method.

* Corresponding author:
amitkiswani@gmail.com (Amit Kiswani)
Published online at http://journal.sapub.org/computer

2. Literature Review

Apple's mobile devices- iPhone and iPad operate on the iOS operating system, Android mobile devices and tablets operate on the Android operating system, and Microsoft's mobile devices operate on Windows operating system. Since 2009, Android and iOS platforms have seen massive success in the number of app installations. The convenience of the mobile phone gave apps an edge over desktop software, and it is increasingly preferred over the mobile web [9]. Hence, it is imperative to know the development platforms for these operating systems.

Native Platform Approach: The Native platform is the 'specific' development approach meaning it requires to write the program logic in the native language of the device itself like in objective C/Swift for iPhone, Java for Android and .net for Windows phone. Advantages of native applications include easy access to operating system and device features like camera, calendar, microphone, etc. It also provides a great user experience in regard to speed since all the operations of the mobile app occurs at the device level. The most prominent drawback with the native approach is

that it is expensive since it requires different skillset of resources and maintenance of the different set of code for different OS.

From document scanning point of view, there are numerous application programming interfaces available that can be integrated into your application. These API's are written in OS-specific language, which would be challenging to integrate and maintain for all platforms.

Cross-platform Approach: Cross-platform is a 'build once, deploy all' solution meaning code is developed once, and it can run on different operating systems like iOS, Android, and Windows. Cross-platform and hybrid platform are the terms used interchangeably. Cross-platform apps are developed in HTML5, CSS, and JavaScript that work in a WebView wrapped in native apps. They are developed with the help of different tools. Tools like Apache Cordova allows the app to access device features like camera, microphone, etc. Most significant advantages of Cross-platform include quick development, easy distribution on the different app stores and is cost effective. The disadvantage of the cross-platform approach is the dependency on the tool and its limitations. Currently, cross-platform end to end scanning solution is very much required since no straightforward solution is available.

3. Challenges for Implementing Scanning Feature

A usual method to implement scanning would be to write the program logic in the native language of the device. The major pro with the native methodology is that the image is not transmitted over the network and scanning can be done on the device itself and finally, the output can be transmitted to the backend server if required. The most significant drawback of native approach is that it is not cost effective and involves more efforts for development and maintenance of the application. The users of the different mobile devices may be using different versions of the operating system, and this makes it difficult for the developer to maintain or provide support. The alternative option is to implement scanning feature on a cross-platform application.

4. Proposed Solution

Cross-platform development is a cost-effective solution, and it is easy to maintain and support one codebase for different mobile operating systems. In this research paper, the solution is formulated using different open source technologies that will support the scanning feature on cross-platform mobile applications. With the cross-platform framework, Apache Cordova, the application can access the device's camera to take a picture or access the photo library for image upload.

Using Apache Cordova on the front-end, the augmented paper image will be captured via device's camera. The document will then be sent to the application server for scanning effects to be applied via computer vision algorithms. Upon the successful scanning results, the document will be saved to the content management system. Document scanning processing occurs on the backend server, unlike Native application. (e.g., Figure 1).

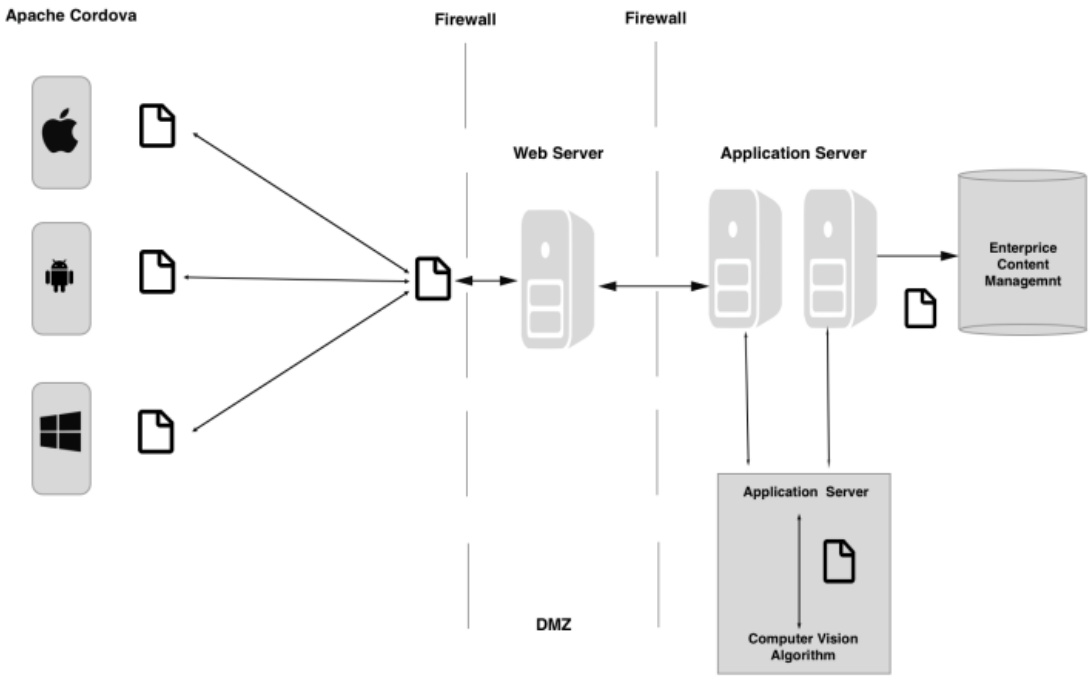

Figure 1. Architecture for cross-platform scanning

5. Cross-platform Mobile Application: Document Upload Steps

5.1. Image Capture via Apache Cordova

Front-end mobile application development will be done on Apache Cordova framework since it is an open source tool and compatible with multiple platforms. It supports a model of plug-in architecture. It allows you to use standard web technologies- HTML5, CSS3, and JavaScript for cross-platform development. Applications execute within wrappers targeted to each platform and rely on standards-compliant API bindings to access each device's capabilities such as sensors, data, network status, camera, etc. [6]. Once the application is built, Cordova camera plugin can be added to the app. Camera plugin gives the application an ability to take a picture, and it provides an option to generate the image in base 64 or binary format [5]. It is recommended to use the binary format since base 64 can increase the size of the image by 30%. Camera plugin provides various methods to accomplish this. [5]

camera.getPicture(successCallback,errorCallback, options)

The camera.getPicture calls the camera application of the respective smartphone device, which allows the user to take a picture of the document; Camera source type must be set to 'Camera' [5]. If a picture is taken successfully then it will pass an image to success Callback function as base 64-encoded String, or as a binary format for the image file (e.g., Figure 2).

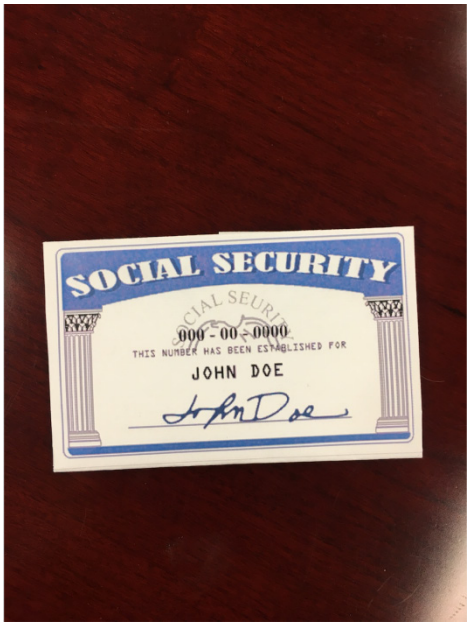

Figure 2. Augmented SSN document image was taken from iPhone 6S camera (Size 2.2 MB) (Width 3024 and Height 4032 pixels)

5.2. Important Option for Camera Settings

Camera.EncodingType: JPEG or PNG

Camera setting is the most crucial thing to provide a primary image for scanning. Most of the mobile devices support two formats of images, JPEG and PNG. When considering the format for document scanning, PNG is more appropriate then JPEG.

PNG vs. JPEG

JPEG is Joint Photographic experts group [2]. It is a bitmap compression format, most commonly used for lossy compression. The compression ratio is adjustable which means you can determine your balance storage size and quality. It uses 8 bits for each red, blue and green that implies every pixel requires 24 bits.

PNG is Portable Network Graphic. An advantage of using PNG is that its compression is lossless meaning there is no loss in quality each time it is opened or saved again. It is excellent for text and screenshots. It also handles detailed high contrast and supports 24-bit RGB and 32-bit RGBA colors. [4]. For these reasons, PNG is good for the textual image and recommended for document scanning.

6. Cross-platform Mobile Application: Computer Vision Scanning Steps

Once the document is captured, it will be transmitted to the backend server. A web service can be implemented that can consume the images, and OpenCV algorithm can be applied to scan these images. Document scanning through OpenCV can be achieved in few steps (e.g., Figure 3). First, detect edges of the paper being scanned, crop the image, and apply some transformative computer vision algorithms to transform an image in scanned like form. The primary challenge with document scanning with smartphones is the lighting condition like where a user would be taking the camera image. Second is the distance at which the user would take the picture. These challenges can be mitigated with OpenCV algorithms explained later in the paper.

Figure 3. Steps for scanning the image

6.1. OpenCV

It is an open source Computer Vision Library. It also offers interfaces in C++, C, Python, and Java and supports different operating systems like Windows, Linux, Mac OS, iOS, and Android. OpenCV is intended for computational capability and with a strong focus on real-time applications. Written in optimized C/C++, the library can take benefit of multi-core processing [1]. For this paper, we will use Java interfaces, as Java is a widely used language with nine million Java developers in the world. Since the release of OpenCV 2.4.4 in Jan 2013, Java binding has been officially developed. The first thing to notice is that OpenCV is a C++ library that should be compiled with operating system specific compiler. OpenCV provides an interface called Java Native Interface (JNI), and to get the native code running in Java Virtual Machine (JVM), one needs to install JNI, this way the native code is required for each platform that your application is going to be run [1]. For example, if you are using windows for development then you have to compile JNI in a windows environment and similarly, if production server is hosted in Linux then JNI need to be generated in Linux environment.

6.2. Basic Matrix

In Computer Vision, we can see the image as a matrix of numerical value, which represents its pixels. For a gray-level image, we usually assign values from 0 (Black) to 255 (White) and the numbers in between shows mixture of both. Each element of the matrices (MAT) refers to each pixel on the gray-level image; the number of columns refers to the image width, and the number of rows refers to the image height. To represent an image in color, we usually adopt each pixel as a combination of three primary colors red, green and blue, so the triplet of colors represents each pixel in the matrix. OpenCV has a variety of ways to describe images signed, unsigned or floating-point data types as well as the different number of channels [1, 3]. For this paper, we will stick with an unsigned integer that ranges from 0 to 255. Several constructions are available for the matrix, for instance:

```
Mat image = new Mat (3024, 4032, CvType.CV_8S);
image = Imgcodecs.imread (pathToFile)
```

The above method will construct the matrix suitable to fit an image with 4032 pixels of width and 3024 pixels of height. And CV_8S represents the 8-bit unsigned integer. The imread method is supplied to get the access to the image through the files Method to load the file into the matrix, make sure you import org.opencv.imgcodecs.Imgcodecs.

6.3. Resize Image

Resizing the images is also one of the essential steps in scanning because it will help to speed up the edge detection process. Resize should be done in the same ratio as the original height and width of the image. Java package, java.awt.Image.*, can be used to reduce the image to required size. For example, we reduced the image to ten times smaller (figure 4).

```
BufferedImage img = ImageIO.read(pathToFile);
Image scaledImg = img.getScaledInstance(newWidth,
newHeight, BufferedImage.SCALE_SMOOTH);
```

6.4. Convert Image to Grayscale and Perform Gaussian Blur

Noise in the image makes edge detection very difficult, so the first step is to remove the noise in the image with a Gaussian filter. Convert the image from RGB to grayscale and then apply the Gaussian filter (e.g., Figure 4).

```
Imgproc.cvtColor(Mat source, Mat gray,
Imgproc.COLOR_RGB2GRAY);
Imgproc.GaussianBlur(Mat gray, Mat blur, new
Size(5.0,5.0), 0.0)
```

Figure 4. Image Outcome after Resize and Gaussian blur (Size 47 kb) (Width 302 and Height 403 pixels)

6.5. Canny Edge Detection

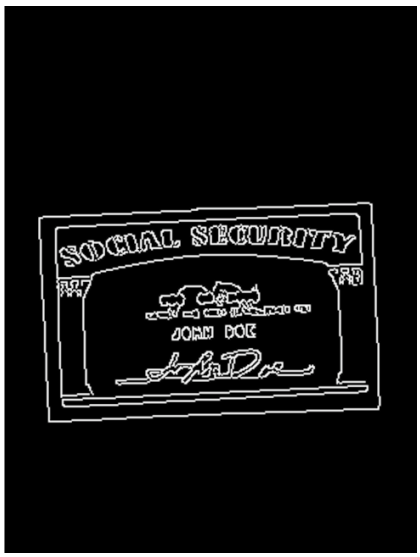

Figure 5. Image outcome after applying Canny

Edge detection is the next step after the Gaussian filter. A canny algorithm is an excellent approach that was proposed by computer scientist John F. Canny, who optimised edge detection for low error rate, single identification, and correct localization. The Canny algorithm applies a Gaussian to filter the noise, calculates intensity gradients through Sobel, suppresses spurious responses, and uses double thresholds followed by a hysteresis that suppresses the weak and un-connected edges. (e.g., Figure 5).

6.6. Finding Contours

Contours are primarily a curve joining all the consecutive points along with the boundary having same color or depth. It is a useful mechanism for shape analysis and object detection and recognition. Finding counter algorithm helps to find the paper edges, that is the X and Y coordinate value of four corners point.

6.7. Perspective Wrapping

Next step here would be to crop the image form the points detected in the previous step of finding contours. The primary method used is warp perspective. Given below is the signature of the method.

```
public static void wrapPerspective(Mat source, Mat
destination, Mat M, Size dsize);
```

The source is naturally the original image taken from the smartphone (e.g., Figure 2), not the resized one. The destination is the outcome image (e.g., Figure 6). Mat M is the wrapping matrix. To calculate it, you can use the getPrespectiveTransform method of class Imgproc. This method will calculate the perspective matrix from two sets of the four correlated 2D points, the source, and the destination points.

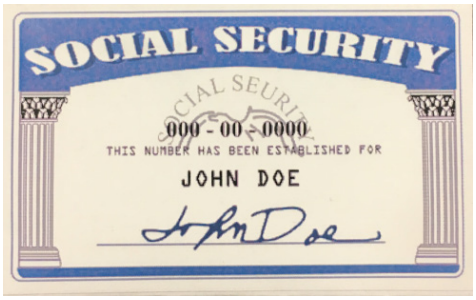

Figure 6. Image after cropping (Size 653 KB)

6.8. Adaptive Thresholding

Once the image is cropped, some Image transformative algorithms need to be executed which will make it lightweight and turn it into black and white. The user can keep it as color if desired but text document in black and white will save a lot of memory storage. One of the most straightforward methods of segmenting an image is using the threshold technique. It will mainly set pixels below a given threshold value as belonging to the interested object as black and the pixels above the threshold value to be converted to white. Another interesting approach to this type of segmentation is related to the use of a dynamic threshold value. Instead of using a given value, the threshold is calculated as a mean of a square block around each pixel minus a given constant. This method is implemented in OpenCV through the adaptive Threshold method, which has the following signature:

```
public static void adaptiveThreshold(Mat source, Mat
destination, double maxValue, int adaptiveMethod, int
thresholdType, int blockSize, double C)
```

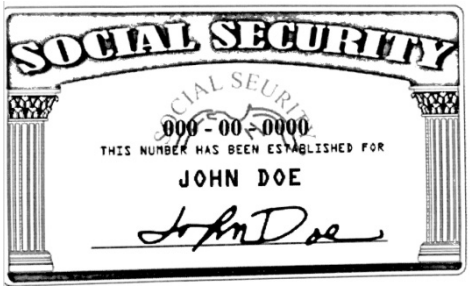

Figure 7. Image after thresholding (Size 274 KB) (Width 1504 and Height 928 pixels)

Note how image size is reduced to half after thresholding. Initial scanned color image size was 635 KB, but the black and white transformed image size is 274 KB. (e.g., Figure 7).

7. Conclusions

Smartphones are handier than the desktops or laptops. Increasing popularity of the cross-platform mobile development approach demands the cross-platform mobile scanner. In this paper, we covered the critical aspects of document scanning, briefly reviewed the limitation of native development approach and presented the cost-effective cross-platform development solution of document scanning. With growing Internet speed (802.11ac standard for Wi-Fi and 4G LTE high network speed), the transmission of image over the network becomes fast which prominently supports the cross-platform approach of document scanning on the backend server. This paper also covered the key aspects of computer vision's algorithms and how they can be useful to read, process and transform digital images. Primary steps for scanning implementation includes a Gaussian filter as well as the essential edge detectors. Conclusive step discussed is image segmenting technique called thresholding with adaptive thresholding results.

REFERENCES

[1] Baggio, D. L. (2015). *OpenCV 3.0 Computer Vision with Java* (Vol. 3). Birmingham: Packt Publishing Ltd.

[2] Designtechnica Corporation. (2017). *PEG vs. PNG: Which image-saving format is the better one to use?* Retrieved from Digital Trends: https://www.digitaltrends.com/photography/jpeg-vs-png-photo-format/.

[3] OpenCV team. (2017). *Home*. Retrieved from OpenCV: https://opencv.org/.

[4] Pew Research Center. (2017, June 28). *10 facts about smartphones as the iPhone turns 10*. Retrieved from pew research: http://www.pewresearch.org/fact-tank/2017/06/28/10-facts-about-smartphones/.

[5] Soko Media. (2017, April 13). *Cross-Platform vs. Native App App Development: Pros and Cons*. Retrieved from www.businessofapps.com: http://www.businessofapps.com/cross-platform-vs-native-app-app-development-pros-and-cons/.

[6] TechSmith Corporation. (2017). *JPG vs. PNG: Which Should I Use?* Retrieved from Tech Smith: https://www.techsmith.com/blog/jpg-vs-png/.

[7] The Apache Software Foundation. (2015). *Cordova Plugin Camera*. Retrieved from Apache Cordova: https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-camera/.

[8] The Apache Software Foundation. (2015). *Overview*. Retrieved from Apache Cordova: https://cordova.apache.org/docs/en/latest/guide/overview/index.html.

[9] Twistfuture Software Pvt. Ltd. (2017, March 17). *Evolution of Mobile Application Development*. Retrieved from www.twistfuture.com: https://www.twistfuture.com/blog/the-evolution-of-mobile-application-development/.