UNIVERSITY OF CALIFORNIA

SANTA CRUZ

THE ELEMENTS OF PROJECT

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Master of Science in Computer Engineering

in

by

Weiting Zhan

June 2019

The Dissertation of Weiting Zhan
is approved:

Professor Roberto Manduchi, Chair

Professor J. J. Garcia-Luna-Aceves

Table of Contents

List of Figures

Abstract

The Elements of Project

by

Weiting Zhan

According to the World Health Organization, it is estimated that globally approximately 1.3 billion people live with some form of vision impairment. In this project, the author developed an IOS application and an Android application, both named ISee, for blind or visually impaired users. Both applications take a picture as input, use Google Firebase MLKit to recognize the character and entities in an image, and then read the text or content of the image to users. ISee used the most advanced Optical Character Recognition and Image Labeling Application programming interface to help blind or visually impaired users get some information about the environment. In order to evaluate the performance of the blur text recognition Algorithm developed by Siyang Qin with ML kit, the author computes the false positive rate and truth positive rate of each of them.

Chapter 1

Introduction

7,675,600 people in the United States reported having a visual disability in 2016, according to the National Federation of the Blind[3]. Globally, it is estimated that approximately 1.3 billion people live with some form of vision impairment according to World Health Organization[4].

1.1 What is Optical Character Recognition and Image Labeling?

Computer stores an image as a stream of raw numbers from a camera sensor, typically an array of color intensities of BGR (Blue, green, Red), as shown in Fig 1.1.

Optical Character Recognition (OCR) transforms images (an array of numbers) into machine-processed text, which is usually encoded using a representing scheme. So we can use a statistic classification method [5][6][7] to separate one group of numbers (for example, all the images of A in the data set) from another

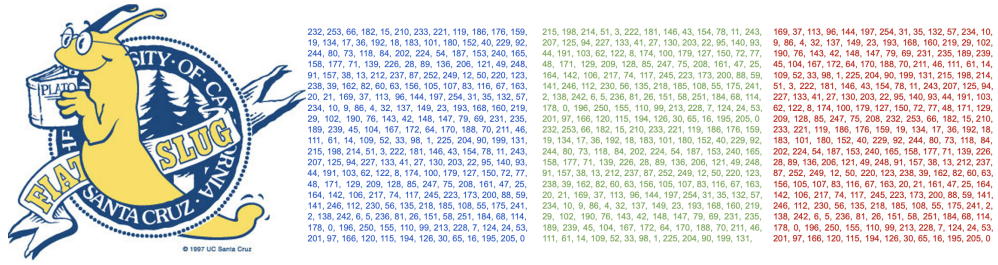


Figure 1.1: Image stored in computer

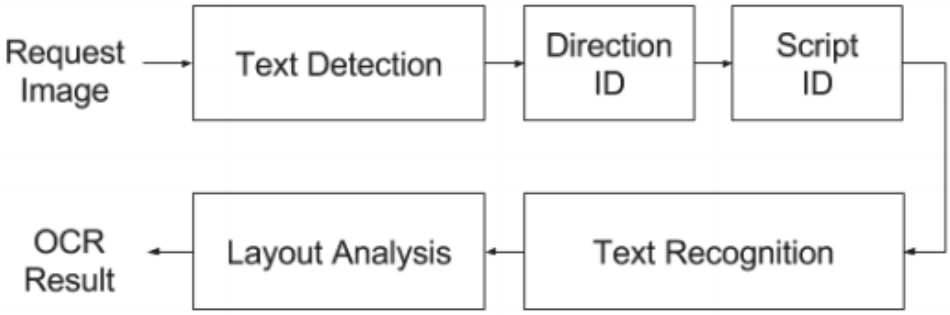


Figure 1.2: Google Cloud Vision API Architecture [1]

group of numbers (all the images of B in the data set).

Google's[1] OCR system, is periodically tested on 232 languages in 30 distinct scripts, achieving state of the art accuracy for most of image types ranging from scanned documents to casual photos.

Label detection[2] predicts the most appropriate labels that describe an image, as shown in Fig 1.4.

Figure 1.3: Google Firebase MLkit text recognition

1.2 Optical Character Recognition with Firebase MLkit

Building a Convolutional Neural Network from scratch and collecting enough data [8] to train the model is a time-consuming and expensive undertaking.

Google Firebase ML kit [9] has APIs for common mobile use cases: recognizing text, detecting faces, scanning barcodes, labeling images, and recognizing landmarks. Other options are IBM Watson Visual Recognition [10], part of the Watson Developer Cloud and Clarif.ai [11]. Here, we use Google Firebase MLkit to build an application for Google's large dataset and computation power.

1.3 Image labeling with Firebase MLkit

Firebase Image labeling [2] gives insight into the content of images. The Image labeling on-device offers

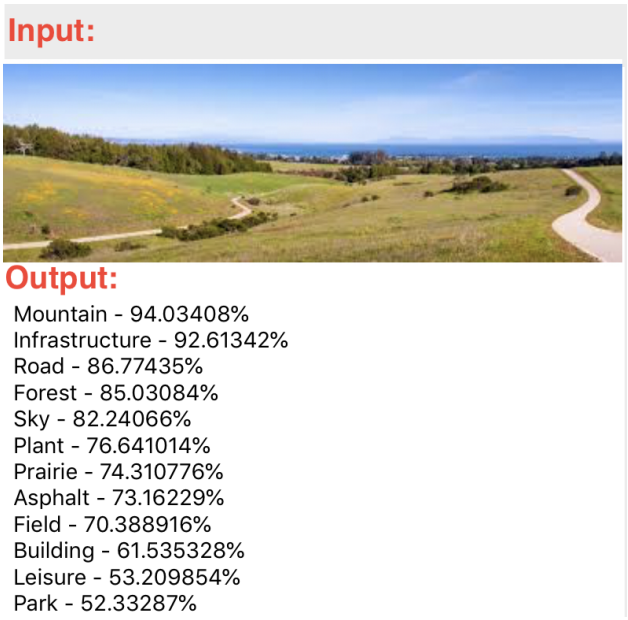


Figure 1.4: Google Firebase MLkit Image labeling demo

	On-device	Cloud
Pricing	Free	Free for first 1000 uses of this feature per month: see Pricing
Label coverage	400+ labels that cover the most commonly-found concepts in photos. See below.	10,000+ labels in many categories. See below. Also, try the Cloud Vision API demo to see what labels can be found for an image you provide.
Knowledge Graph entity ID support	✓	✓

Figure 1.5: Google Firebase MLkit Image labeling API[2]

400+ labels that cover the most commonly-found concepts in photos, and the cloud offers 10,0000+ labels in many categories, as shown in Fig 1.5

1.4 Text to Speech API

With the release of iOS 7, Apple introduced a text to speech API, AVSpeechSynthesizer class[12], that allows developers to add text to speech functionality to an application quickly and easily.

Android Text to speech (TTS)[13] allows developers to add text to speech, and Android TTS was available from version 1.6.

1.5 Blur Text recognition Algorithm

The Blur Text recognition algorithm [14] is developed by Siyang Qin. In the project, the author compared the result of MLkit OCR with work from The Blur Text recognition algorithm [14].

Chapter 2

Related Work

2.1 Apple Store

In the Apple store, ***Talking Googles - a camera with speech*** can recognize images and say what it finds. It can recognize almost any logo, landmark, book, product, artwork, text. It has a video model that continuously checks the video stream for identities and says them. ***ThatHelps-Find ways to help*** serves to connect the volunteer with a blind user. ***Be My Eyes - Helping the blind*** is also a social network connects volunteer to blind user.***Seeing AI***, developed by Microsoft, uses the phone's camera to read what it sees automatically. It can read short text,document,product,person,currency,scene,color,handwriting to user. ***BlindWays:bus stop navigation***. help guide blind user to within a cane's distance of an outdoor bus stop sign using permanent landmark clues contributed by volunteers.***BeSpecular***, volunteer help blind by listening to the question sent by blind, or looking at the pictures, and reply with a friendly voice note or text message.***Read for the Blind*** create audio books for the blind, by reading books or short articles from magazines, newspapers

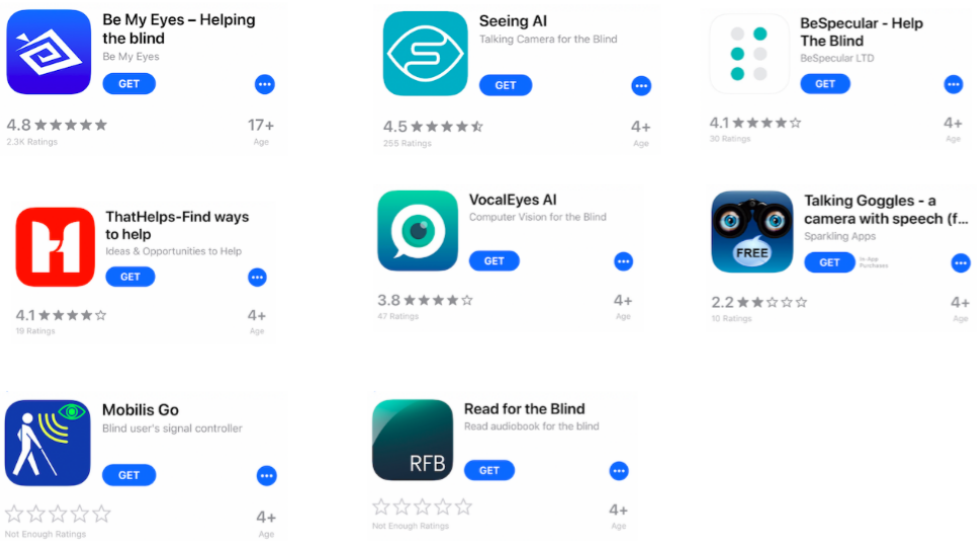


Figure 2.1: Application for blind user in Apple Store

or any interesting websites.***VocalEyes AI, computer vision for the blind***, created at MIT, identify objects, food, animals,currency and products, as well as read text, detect faces, recognize emotions and describe environments using AI. ***Mobilis Go*** is designed for the blind or visually impaired user. The application uses Bluetooth LE protocol to activate beacons. Activates sound beacons located on objects (like doorways, traffic lights.) Beacons emit a sound to help the user to navigate.

2.2 Google Play Store

In the Google Play store, there are 3 types application for blind user: First, connect a blind user with volunteers.***Be My Eyes -Helping the blind***, enables blind to receive live assistance from sighted volunteers. ***BeSpecular-Help the Blind***, connect volunteers to help the blind user to identify objects, situations or images. Second, navigation application.***Lazarillo GPS*** for Blind use audio messages to tell the visually impaired user about nearby places, the street walking on. ***Blind Assistant Free*** turns the phone into a sonar

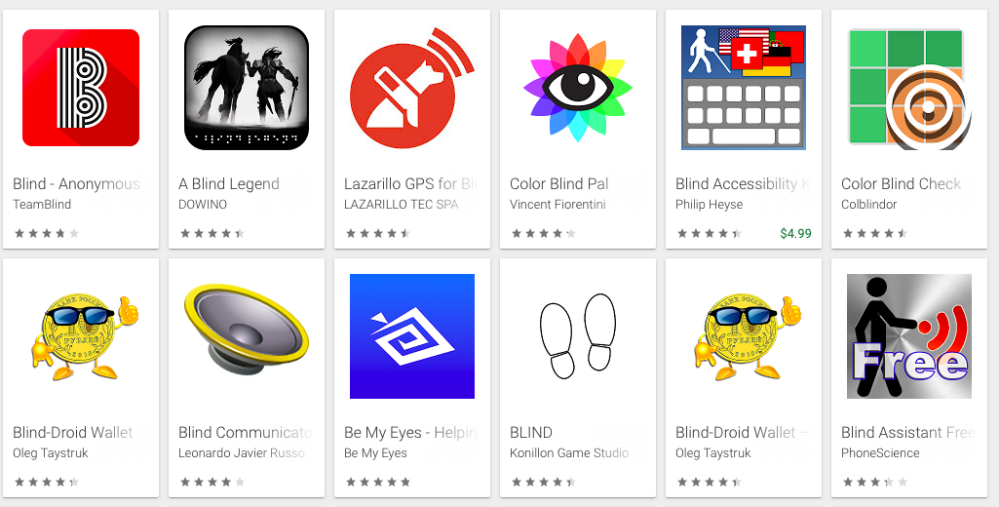


Figure 2.2: Application for blind user in Google Play Store

employing a speaker to emit the sound pulses and a microphone to record the return echoes to detect the presence of nearby obstacles and estimate the distance. ***GetThereGPS nav*** for the blind, does not display a map, but tells where is the user and how to get to the destination. ***Blind Explorer*** is a senatorial guidance system that integrated 3D sounds and satellite navigation technologies to navigate. Third, identify entity for blind user.***VIP Code Reader***, can detect any QR code or barcode that is present in the camera's field of vision. **Blind-Droid Wallet** is an offline paper money recognition.***Blind Communicator*** has a voice guide that tells the user everything that it's happening in the device. ***NowYouSee—A colorfull world for the color blind***, created s precisely for the type of color blindness.

In conclusion, there are only 3 applications ***Seeing AI***,***Talking Googles - a camera with speech*** and ***VocalEyes AI, computer vision for the blind***, in Apple Store and Google Play Store that designed to describe the environments or to surround to them just by taking a picture of the environments. The API they used is unknown. This project offers more options for a blind user to choose from.

Chapter 3

Experiment and Result

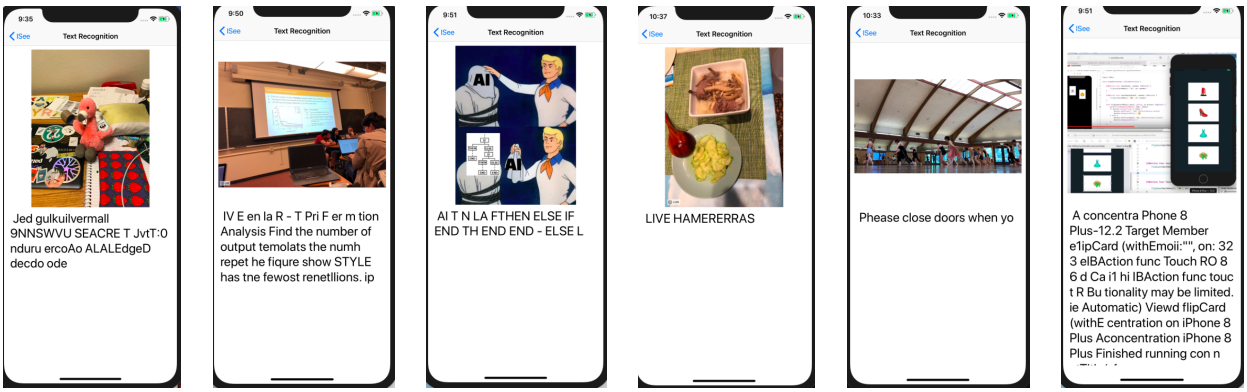
Since the application is built on Google Firebase MLkit, the accuracy of the ISee depends on the Google Firebase MLkit, which based on the largest dataset, most advanced algorithms and sufficient computing power.

3.1 ISee on IOS device

3.1.1 Optical Character Recognition

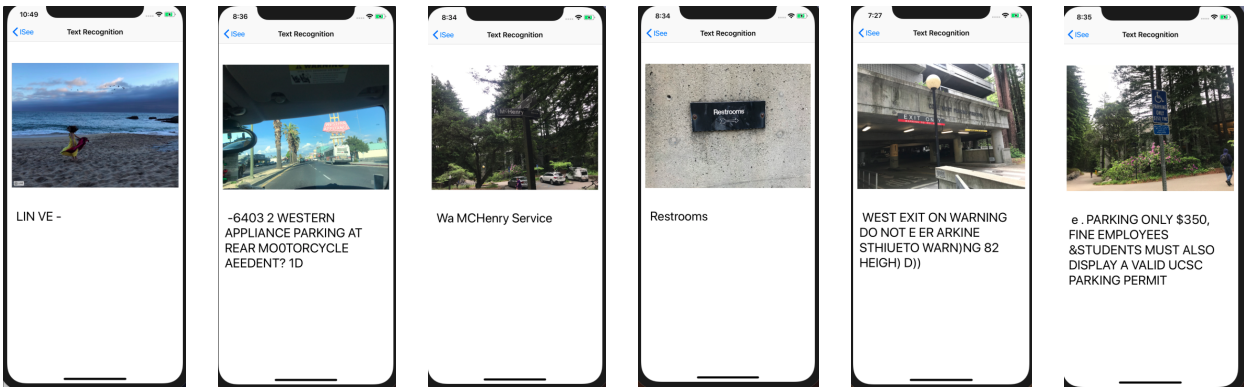
In order to test ISee's optical character recognition feature on IOS device, the author uses 6 indoor images and 6 outdoor images as input, and the output is shown in Fig 3.1 and Fig 3.2.

The OCR API show some solid result for text recognition. For example, in Fig 3.1 (e) Dancing Studio, the app recognized "Please close doors when", it's hard for a regular user to notice the text in the image. Fig



(a) Desk (b) Presentation (c) Comic (d) Food (e) Dancing Studio (f) Screen Shot

Figure 3.1: Indoor Image Text Recognition on IOS device



(a) Beach (b) Road (c) signpost (d) Library Sign (e) Parking lot (f) Parking sign

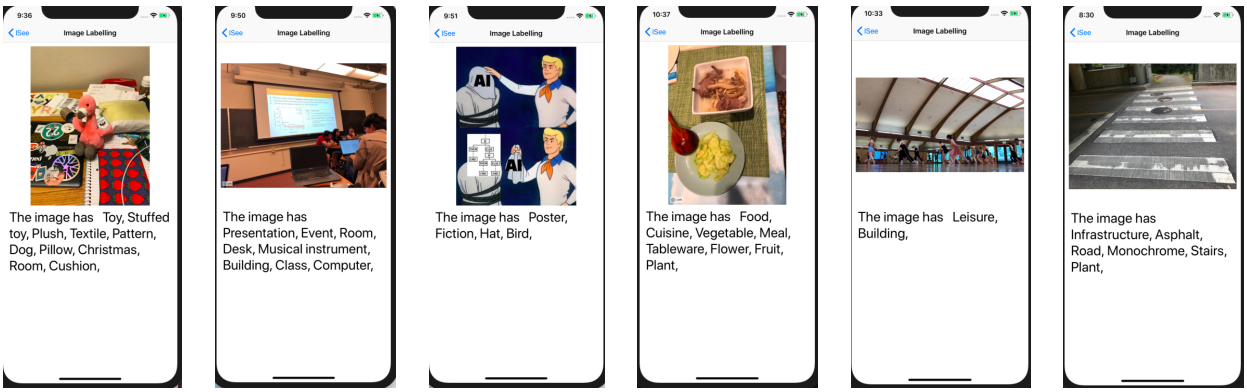
Figure 3.2: Outdoor Image Text Recognition on IOS device

3.2 (f) Parking sign, gives the useful information in the image. However, The OCR API not robust enough for blind users. First, it will return misleading information. For example, As shown in Fig 3.1 (c) Comic, the API mistake ”IF” for ”T”. Second, the API didn't check the spell of the word. It will recognize some no meaning words, such as ”temolats” in Fig 3.1(b)Presentation, ”AEEDENT” in Fig 3.2 (b)Road and ”E Er ARKINE” in Fig 3.2 (e)Parking lot.

3.1.2 Image labeling

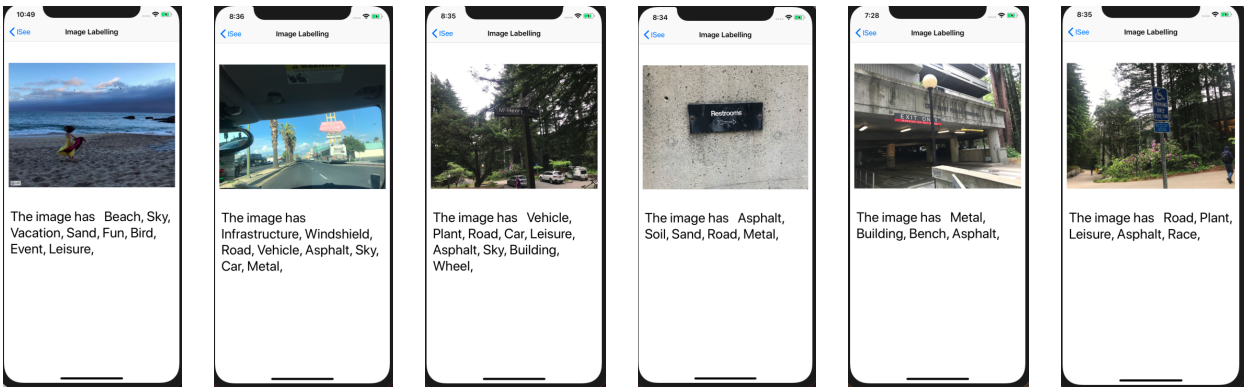
In order to test ISee's Image Labeling feature on IOS device, the author uses 6 indoor images and 6 outdoor images as input, and the output is shown in Fig 3.3 and Fig 3.4.

The Image labeling API gives more useful information that the blind user may interest in. In Fig 3.3 and Fig 3.4, the application can give partly correct information. Fig 3.3 (a) Desk, has ”toy”, (b) Presentation has”presentation, event, room, desk, class, computer” Fig 3.4 (a) Beach has ”Beach, sky, sand, bird”, and (c) signpost has ”vehicle, plant, road, car, building,wheel”. At the same time, the API makes some interesting mistakes. First, it adds something that didn't exist in the image, such as Fig 3.3 (b) Presentation, no musical instrument exists. (c) Comic ”Bird” and so on. This may give misinformation to the blind user. Second, the API missed some distinct entities in the image. For example, people in Fig 3.4 (a) Beach and (e) Dancing Studio. People are important for blinder users. However, face recognition is separated from image labeling API.



(a) Desk (b) Presentation (c) Comic (d) Food (e) Dancing Studio (f) zebra crossing

Figure 3.3: Indoor Image Image Labeling on IOS device



(a) Beach (b) Road (c) signpost (d) Library sign (e) Parking lot (f) Parking sign

Figure 3.4: Outdoor Image Image Labelling on IOS device

3.1.3 Text to Speech

To test the text to speech feature on IOS device, the author uses the result from Optical character recognition or image labeling, add a template to the result.

For optical character recognition, if the text is detected, the template will be ”the text in the image are + result Of OCR”. Otherwise, the template will be ”I am unable to detect any text in the image.”

For image labeling, the template will be ”The image has” + result from Image labeling +” in the image.

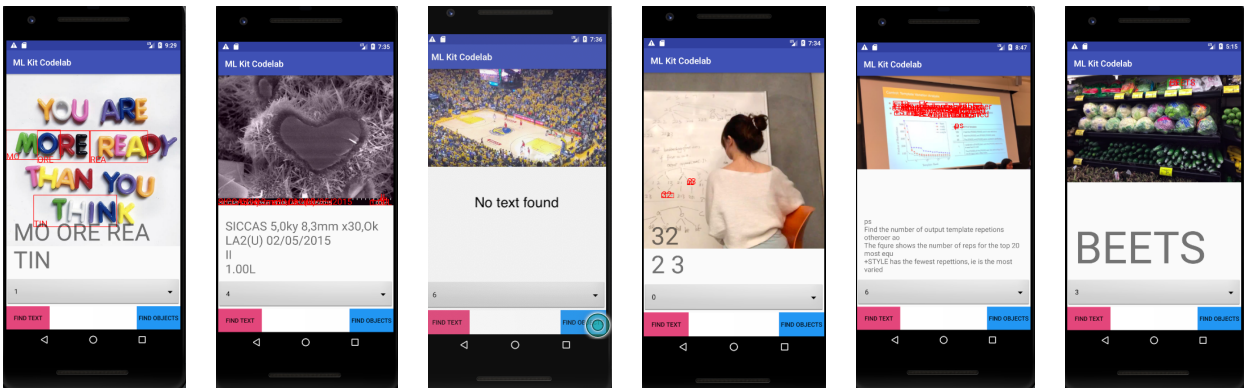
Test 6 images from Optical Character Recognition and 6 images from Image labeling. The template works fine, and the text to speech works fine.

3.2 ISee on Android device

3.2.1 Optical Character Recognition

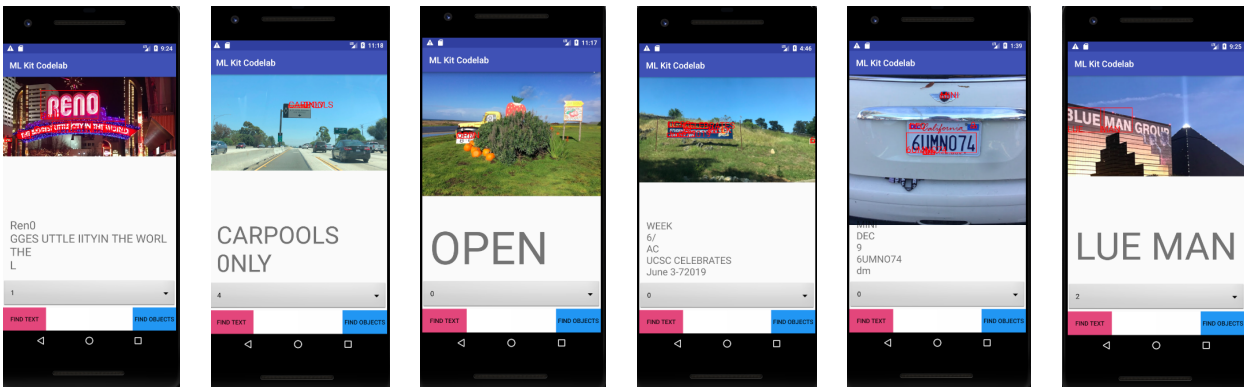
The application is based on Firebase MLkit. In order to test ISee's optical character recognition feature on Android device, the author uses 6 indoor images and 6 outdoor images as input, and the output is shown in Fig 3.6 and Fig 3.6.

The Text recognition API show some amazing result in Fig 3.6 (b)Carpools sign, (c) Farm and (e) License plate. It gives important and correct information in the image. However, the text recognition API didn't work well on handwriting Fig 3.5 (d) and curled words 3.6 (a). And couldn't recognize blur text in Fig 3.5 (c).



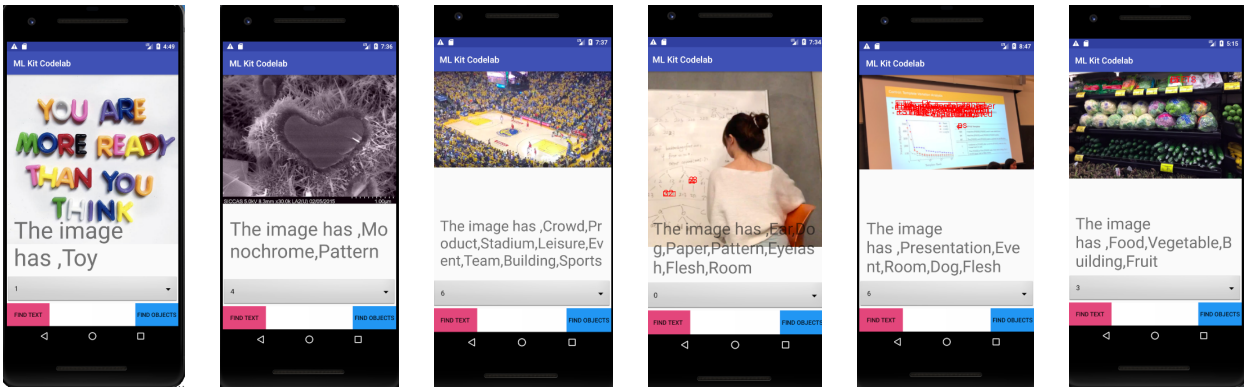
(a) Candy letters (b) SEM image (c) NBA Oracle Arena (d) Handwriting (e) slides (f) Super market

Figure 3.5: Indoor Image Text Recognition on Android device



(a) Reno sign (b) Carpools sign (c) Farm (d) UCSC sign (e) License plate (f) Building

Figure 3.6: Outdoor Image Text Recognition on Android device



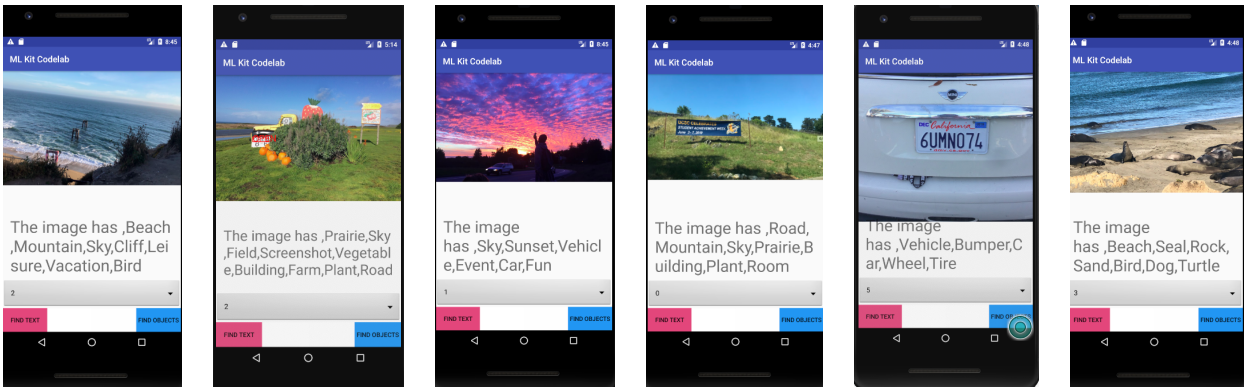
(a) candy letters (b) SEM image (c) NBA Oracle Arena (d) Handwriting (e) Slides (f) Super market

Figure 3.7: Indoor Image Image Labelling on Android device

3.2.2 Image labeling

In order to test ISee's optical character recognition feature on Android device, the author uses 6 indoor images and 6 outdoor images as input, and the output is shown in Fig 3.7 and Fig 3.8.

The image labeling API gives more useful information of the image. Such as "crowd, event, Team, Sports" in Fig 3.7 (c) NBA Oracle Arena. "Presentation, event" in Fig 3.7 (e) Slides , "Food, vegetable" in Fig 3.7 (f)Super market, "Beach, sky ,cliff, Bird" in Fig3.8(a) Beach, "Sky, Field,vegetable, Farm, Road" in Fig3.8 Farm, "Sky, sunset, vehicle, car" in Fig3.8 (c) Sunset and so on. The API also gives misinformation such as no "wheel, tire" in Fig3.8 (e)License plate and no "dogk eyelash" in Fig 3.7 (e) Slides.



(a) Beach (b) Farm (c) Sunset (d) UCSC sign (e) License plate (f) Beach

Figure 3.8: Outdoor Image Image Labelling on Android device

3.2.3 Text to Speech

To test the text to speech feature on Android device, the author uses the result from Optical character recognition or image labeling, add a template to the result.

For optical character recognition, if the text is detected, the template will be ”the text in the image are + result of OCR”. Otherwise, the template will be ”I am unable to detect any text in the image.”

For image labeling, the template will be ”The image has” + result from Image labeling +” in the image.

Test 6 images from Optical Character Recognition and 6 images from Image labeling. The template works fine, and the text to speech works fine.

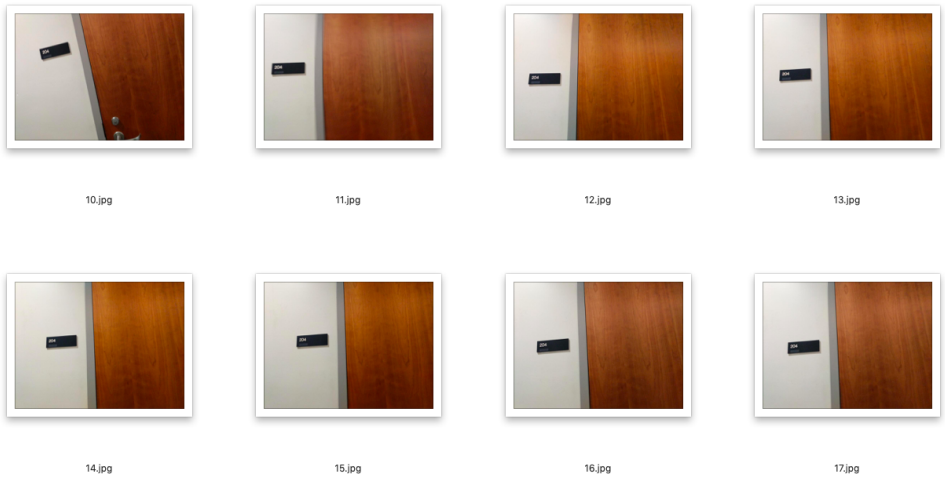


Figure 3.9: Test image

3.3 MLkit OCR vs Blur Text Spotter

In order to evaluate the performance of the blur text recognition Algorithm developed by Siyang Qin with ML kit, the author computes the false positive rate and truth positive rate of each of them.

3.3.1 Data

There are 939 images, as shown in Fig 3.9, labeled with ground truth. In the ground truth data, if an image is labeled as 'Skip,' it does not contain text. Otherwise, it contains text.

The heat map Images generated by Blur text Spotter are shown in Fig 3.10. The white pixels in heat map images mean siyang's algorithm spots text. The black pixels in heat map images means no text spotted by siyang's algorithm. There are 552 heat map images with ground truth.

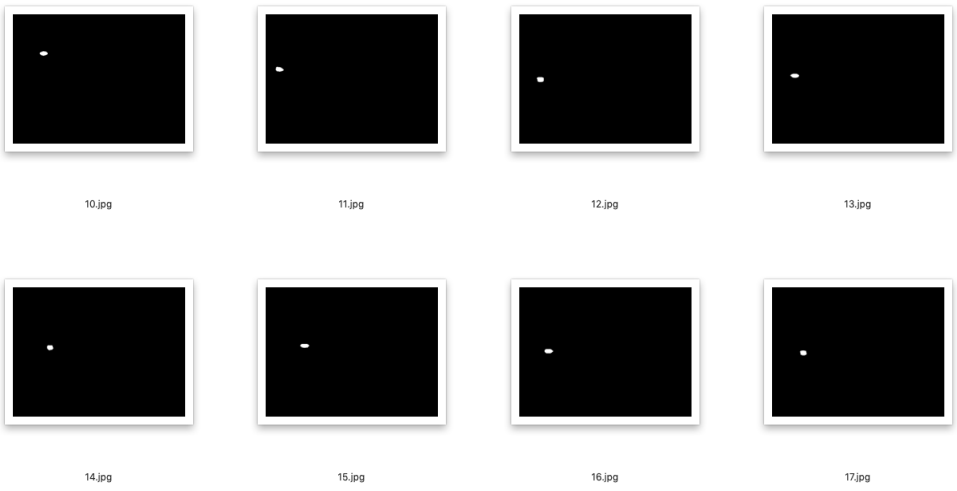


Figure 3.10: The heat map images of Blur text Spotter

3.3.2 Experiment

The code for the experiment can be found:https://github.com/WaitingZhan/MLkit-vs-Siyang

MLkit experiment

The author uses MLKit API processed 939 images with ground truth and generated the result of MLKit. In the result of MLkit, ’0’ means no text found by MLKit API for that image; otherwise, the connect is the text found by MLKit API.

For each image, a False Positive is defined as when MLKit finds text and the ground truth label is ’Skip’. A True Positive is defined when MLKit finds text and the label is not ’Skip’.

True Positive Rate(TPR) is defined as:

$$TruePositiveRate = \frac{The number of True Positive in the dataset}{The size of the dataset}$$

False Positive Rate(TPR) is defined as:

$$FalsePositiveRate = \frac{The number of False Positive in the dataset}{The size of the dataset}$$

Siyang's algorithm experiment

For siyang's algorithm, the text were found when the number of largest connected white pixels is greater than the number of N. The range of N is defined by the range of the number of largest connected white pixels.

For each heat map image, a False Positive is defined as when siyang's algorithm finds text and the ground truth label is 'Skip'. A True Positive is defined as when siyang's algorithm finds text and the label is not 'Skip'.

In the range of N, the author calculate True Positive Rate(TPR) and Faulse Positive Rate(FPR) for each N, and then generated the Receiver Operating Characteristic (ROC) curve.

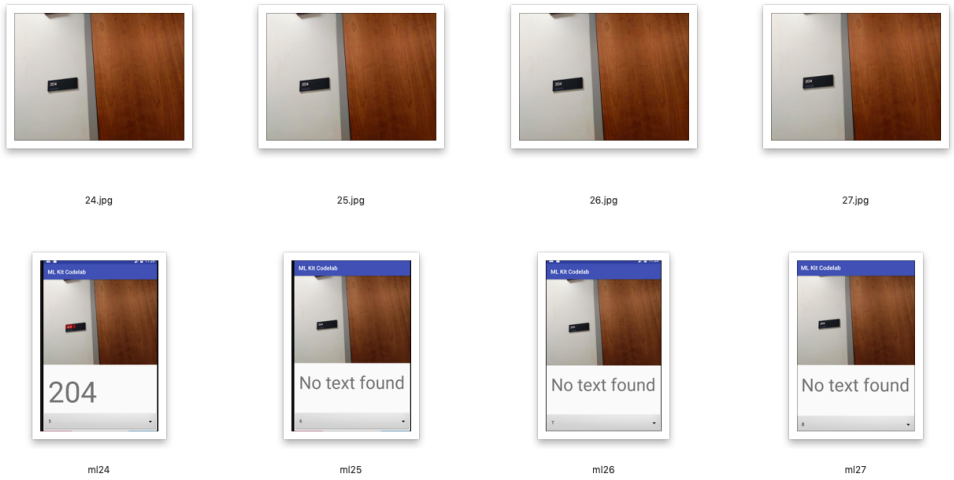


Figure 3.11: ML kit text recognizer result

3.3.3 Result

ML Kit OCR result

The author tests 910 images with labeled ground truth on MLkit. There are 53 images when MLKit finds text and the ground truth label is 'Skip'. There are 843 images when MLKit finds text and the label is not 'Skip'.The false positive rate (FPR) for MLkit Text recognition is 0.057, and the true positive rate (TPR) for MLkit is 0.942. Here is some example from MLkit shown in Fig 3.11. The MLkit is unable to detect the blurred image, only identify 1 out of 4 images with the number "204".

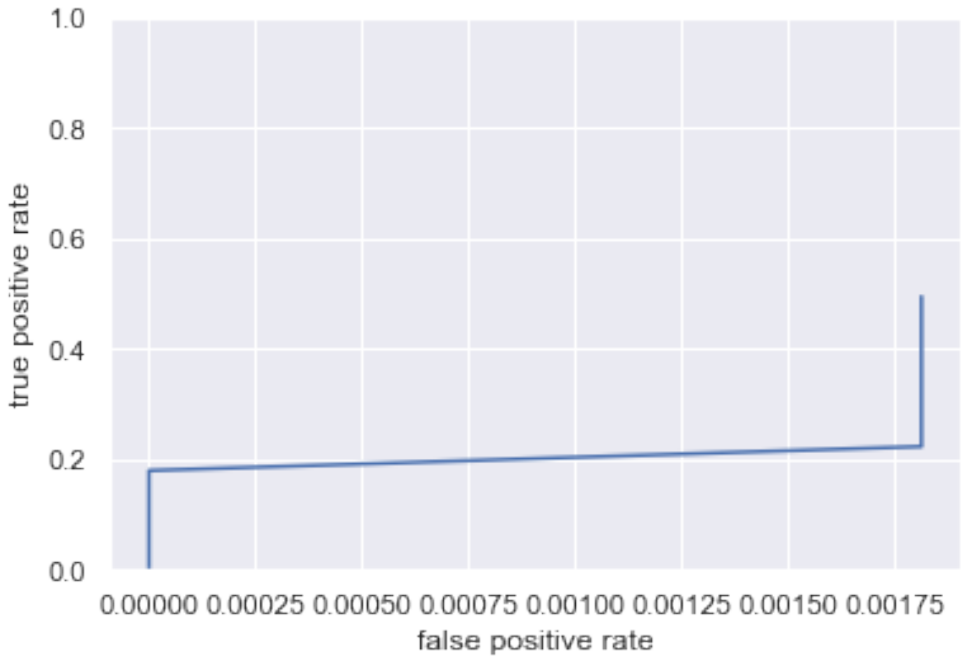


Figure 3.12: Receiver Operating Characteristics for siyang's Algorithm

Blur Text Sport result

552 images have a heat map and ground truth. Among these 552 images, only 1 image with ground truth "skip" has at least 1 connected white pixel (24 pixels). N is range from 1 to 162. The ROC curve is shown in Fig 3.12.

Chapter 4

Conclusion

The project made an ISee application for blinder user or visually impaired user on IOS device and Android device. ISee can read the text and the entity in the image for a blind user. The accuracy of the ISee depends on Google Firebase Text recognition API and Image Labeling. The performance of ISee is not enough for navigation, just offer a glimpse of the real world to a blind user. The author also computes the false positive rate and truth positive rate of MLkit's text recognition and blur text recognition Algorithm developed by Siyang Qing. Using 939 images with labeled ground truth, the FPR for MLkit is 0.057, and TPR for MLkit is 0.942.

Appendix A

Compiling

The code for the project can be found:https://github.com/WaitingZhan/ISee.

The code for comparison between ML kit text recognizer and siyang's blur text algorithm can be found: https://github.com/WaitingZhan/MLkit-vs-Siyang.

Prerequisites for firebase:

A.1 System Prerequisite for IOS

iOS 11.4

Xcode 10.1

Swift 4.2

Firebase

AVSpeechSynthesizer

A.2 System Prerequistie for Android

Android Studio Version 3.4.1

Java SE 10

Firebase Targets API level 16 (Jelly Bean) or later Uses Gradle 4.1 or later

Text to speech

A.3 MLkit Text recognition vs Siyang's Algorithm

Python 3.7.1 Jupyter notebook

Bibliography

[1] Google document text tutorial. https://cloud.google.com/vision/docs/fulltext-annotations.

[2] Firebase image labeling. https://firebase.google.com/docs/ml-kit/label-images.

[3] United states blindness statistics. https://www.nfb.org/resources/blindness-statistics.

[4] World health organization blindness statistics. https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment.

[5] Sotiris B Kotsiantis, Ioannis D Zaharakis, and Panayiotis E Pintelas. Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review*, 26(3):159–190, 2006.

[6] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24, 2007.

[7] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015.

[8] Google datasets. https://ai.google/tools/datasets/.

[9] Google mlkit. https://firebase.google.com/products/ml-kit.

[10] Ibm classifier. https://www.ibm.com/watson/services/visual-recognition/.

[11] clarifai.com. https://clarifai.com.

[12] Ios avspeechsynthesizer class. https://code.tutsplus.com/tutorials/create-a-text-to-speech-app-with-swift--cms-22229.

[13] Android text to speech tutorial. https://javapapers.com/android/android-text-to-speech-tutorial/.

[14] Siyang Qin and Roberto Manduchi. A fast and robust text spotter. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8. IEEE, 2016.

Acknowledgments

The author, thanks to Professor Roberto Manduchi and for his feedback and helpful suggestions.

The author thanks Hyein Jeong for the data collection in the project.

The author thanks Suzanne da Camara, and Bill for grammar checking.

The author thanks Professor J. J. Garcia-Luna-Aceves for reading the report.

The author thanks sifang you.