OPITICAL CHARACTER RECOGNITION WITH TRANSLATION AND LASER DETECTION

On Sang Anson Law

Dpt. Of Computer Science and Engineering
The Chinese University of Hong Kong

August 24, 2018

Abstract

With the ease of travel in the modern day, traveling to a foreign country is very common. When traveling to foreign countries, there is a high chance that they may not speak your language and you do not speak theirs. To overcome this communication problem, I have created an application that uses computer vision and OCR to translate written words then using a text-to-speech system to output the translated word. Although this program does what it is designed to do, there are still inaccuracies with the word translations, and the efficiency of the program can be improved.

Table of Contents

Introduction

Since the start of time, communication has been a major part of how humans interact with each other, but it has its own limitations. For example, if we are in a foreign country on holiday and we are trying to order food, but the menu is in a language which we don't understand, what do we do? Of course, sometimes you may have a waiter to help us translate, but that may not always be the case.

I have decided to create a program that would ideally be used for translating small words and phrases, such as on menus. The idea behind this program is there would be a laser and a camera; the laser emits a red dot onto a word, then the camera is about to track where the laser is, and based on the position of the laser extract the word that is closest to it. After extracting the word, the word would be translated into a desired language chosen by the user and the word would be outputted through some speakers.

The main challenges required to overcome in-order to build this program is the tracking of the laser and extracting the word near the laser. The main concept behind tracking the laser would be to use the HSV properties on an image, and apply filters to it so that only the red (assuming a red laser is used) from the laser is left. By removing the unnecessary details on the image and leaving only the laser, we would then be able to calculate the middle point of the patch of red and hence find the mid-point of the laser. After acquiring the position of the laser, the word extraction would be to use an OCR software to locate the words that are on the image and using basic geometry to work out which word is closest to the laser.

The rest of the papers would be as follows: section 2 contains the background and recent related projects of computer vision and OCR; section 3 contains the theory behind my program, the algorithm that I have used and why I have decided to make certain choices; section 4 contains the implementations and experimental results, such as problems that has occurred when using the pre-designed algorithm and what I did to overcome such problems, and also the limitations of my program; section 5 contains the discussion of how the program has done overall, the limitations it may have compared to other projects; section 6 contains a formal conclusion of this project.

Background

History of computer vision and OCR

Computer vision is a way in which scientist have tried to use technology to replicate human vision. In the late 1960s, some universities have started to research into artificial intelligence (AI), they were hoping that through AI they can impersonate the human visual system so that it can be implemented on robots for intelligent behaviour [1]. By the 1970s, the foundations of computer vision algorithms has been created; for example, extracting edges from image to recognise objects, labelling of lines and representing objects as interconnecting smaller structures [1]. OpenCV is an open source project developed by Intel in 1999 for real time computer vision. Nowadays, OpenCV is commonly used for computer vision related programs, this is because OpenCV is open source so that developers are able to use the software freely; furthermore, even though it is mainly created to be used for C++, there are bindings for Python, Java and MATLAB allowing a larger audience to be able to user it.

Optical Character Recognition or OCR is the way in which handwritten or printed text is converted into machine encoded text. Many machine processes require OCR, some examples include; text mining, machine translation, and text to speech, the latter two we will be using in this problem. Tesseract is an open source OCR software since 2005 after Hewlett Packard released it to the public and sponsored by Google from 2006. As Tesseract is open source, it is also used in many applications; furthermore, Tesseract is able to support a large abundance of languages such as: common Western languages (French, German, Spanish etc.), ideographic languages (Chinese, Japanese), and also languages that read form right-to-left (Arabic, Hebrew).

Related Projects

Over the years, there have been many papers and research done on optical character recognition. The earliest traceable papers are by Edmund Fournier d'Albe who developed the Optophone in 1914. This is a device that is able to scan a across a printed page and produces tones for the recognised characters and symbols [2].

More recently, because of technological advancement, OCR translation can be applied in many scenarios. For example, there is research into implementing OCR translation on augmented reality (AR). In this research by Lamma Tatwant and Henda Chorfi Ouertani, they are trying to find out whether it is plausible to use modern mobile phones as a translation device through AR. The main function of OCR was for text extraction after collecting images from the phone's camera [3]. This research is similar to what we are trying to achieve here, by collecting a stream of images and translating the words from that stream; however, the authors use AR to provide the output of the results, whereas we are using a speech-based output.

Another similar example using the application of OCR and AR was developed by Mahesh Bhargava et al.. It uses the phone's camera to capture an image, then provides the translation of the detected words on top of the words in the captured image. Furthermore, after the translated words have replaced the words on the original image, a text-to-speech module would be used to read out the translated words. To develop this application, they have used Tesseract OCR for extracting text from the images and converting into machine encoded text. [4]

Theory

The technological advancements of recent years have allowed us to put more improved processors into smaller devices, making mobile devices faster and more powerful than ever before. This increase in speed and processing power has allowed the use of OCR in our project.

OCR is the key to making our project work, therefore I have chosen to use Python as the language to write this program in. Python has a vast number of libraries already available for scientific computation like Numpy [5], or Tesseract for OCR. Furthermore, Python, as a high-level language, is much more readable and easier to understand as opposed to a language like C or C++ where someone not used to that language may struggle. Even though there may seem to be a lot of advantages to using Python there are still some drawbacks; for example, a program programmed in Python may be slower to run compared to a program written in C or C++, due to the code in Python being interpreted at runtime whereas C is compiled before it is run [6].

The main focus in creating this program is the tracking of the laser pointer - getting the co-ordinates of the laser on our camera, then using the co-ordinates to find the word that is closest to that coordinate and translate it. There is a Python program created by Brad Montgomery where he is able to successfully use OpenCV to track a laser pointer from a live video stream [7]. The main workflow of the program as shown in Figure 1 would be an implementation using his pre-built program. The workflow of my program is as follows:

1. Using OpenCV, get a capture a video frame with the camera
2. Find the laser on that frame
3. Find and translate the word closest to the laser point given the co-ordinates of the laser point
4. Use Python's audio module to play translated word
5. If there is no quit key detected repeat from step 1, if not then quit the program

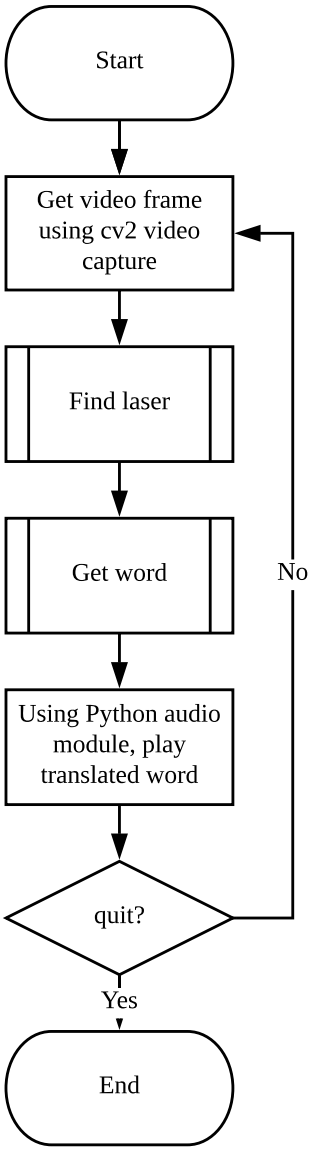


Figure 1

I have decided to separate step 2 and 3 into separate parts, as they are the two main focuses of the algorithm. To successfully track the laser using OpenCV in Python I used a program written by Brad Montgomery [7], the algorithm of this program as shown in Figure 2 is as follows:

1. Convert the video frame to HSV using OpenCV's cvtColor function
2. Split the frames into individual HSV components with OpenCV's split function
3. Apply a threshold to each of the components so that it filters out all the colours except the red from the laser
4. Apply an AND operation on the three components to cut down the false positives and merge the channels back together
5. Using the new merged image, apply OpenCV's findContours function
6. Using the largest contour found with the findContours function, find the mid-point of that contour
7. Return the mid-point

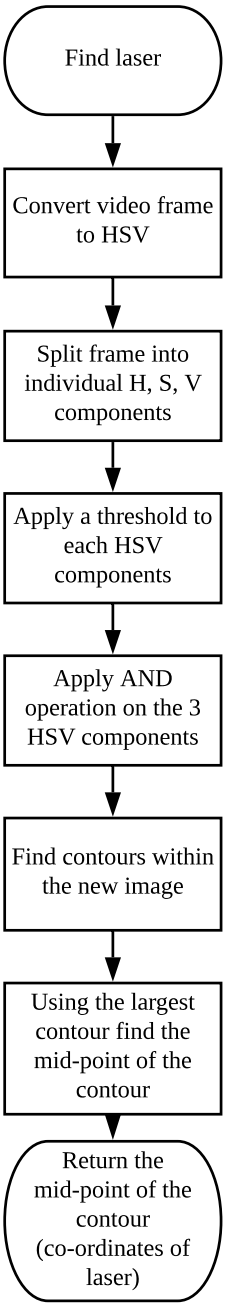


Figure 2

The algorithm for finding the word and translating it as shown in Figure 3 is as follows:

1. Using the co-ordinates of the laser, crop the image leaving only the surroundings of the laser. This is to reduce the number of words needed to be checked by Tesseract.
2. Concurrently
    a. Using Tesseract, find the bounding boxes for the words in the cropped image
    b. Also using Tesseract, find the words in the cropped image
3. Match the words found with Tesseract with the bounding boxes

4. Find the word that is closest with the laser, using the mid-point of the bounding boxes
5. Translate the word found using Python's googletrans module
6. Return the translated word

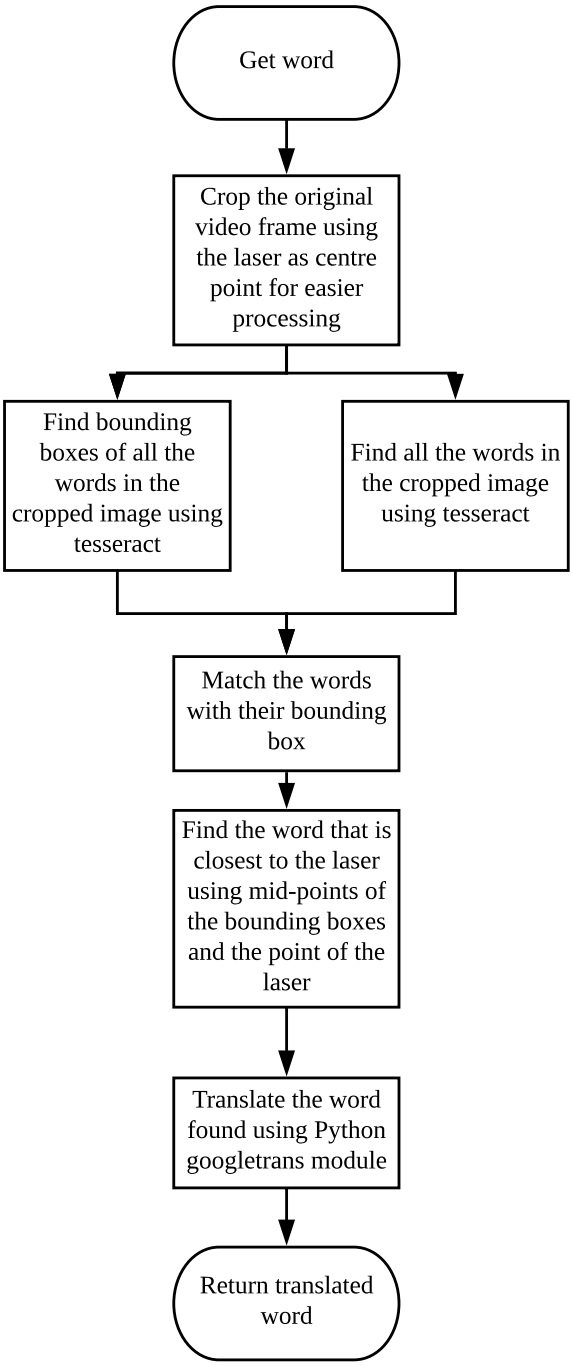


Figure 3

As you can see from Figure 3 that finding the bounding boxes and the words are done concurrently. I decided to do this because both processes, in theory, would take a similar amount of time. It is also the most time-consuming operation of the program so therefore, by running the two methods concurrently I can cut down the operation time by nearly half compared to running the methods linearly.

Implementation and Experimental Results

For experimental purposes, I have created the program in Python, and tried running it on a 2017 MacBook Pro running a 2.8 GHz Intel Core I7 processor (7920HQ). In the experiment I used a pre-printed set of words on a piece of paper, using the laptop's camera to do the video capturing and recording.

For the development of this program I first developed the function for finding and extracting words on a piece of paper. To test this function, I used a set of images, each image containing a variable number of words and lengths. As stated before, for this function I decided to run some methods concurrently in order to decrease the time it would take to run the program. In Figure 4, it shows the time difference between programming the program to run linearly and concurrently.

Figure 4

Linear time (s)	Concurrent time (s)
5.0078	2.3776

The data in Figure 4 is produced by feeding the program the image in Figure 5, producing the result in Figure 6.

Figure 5

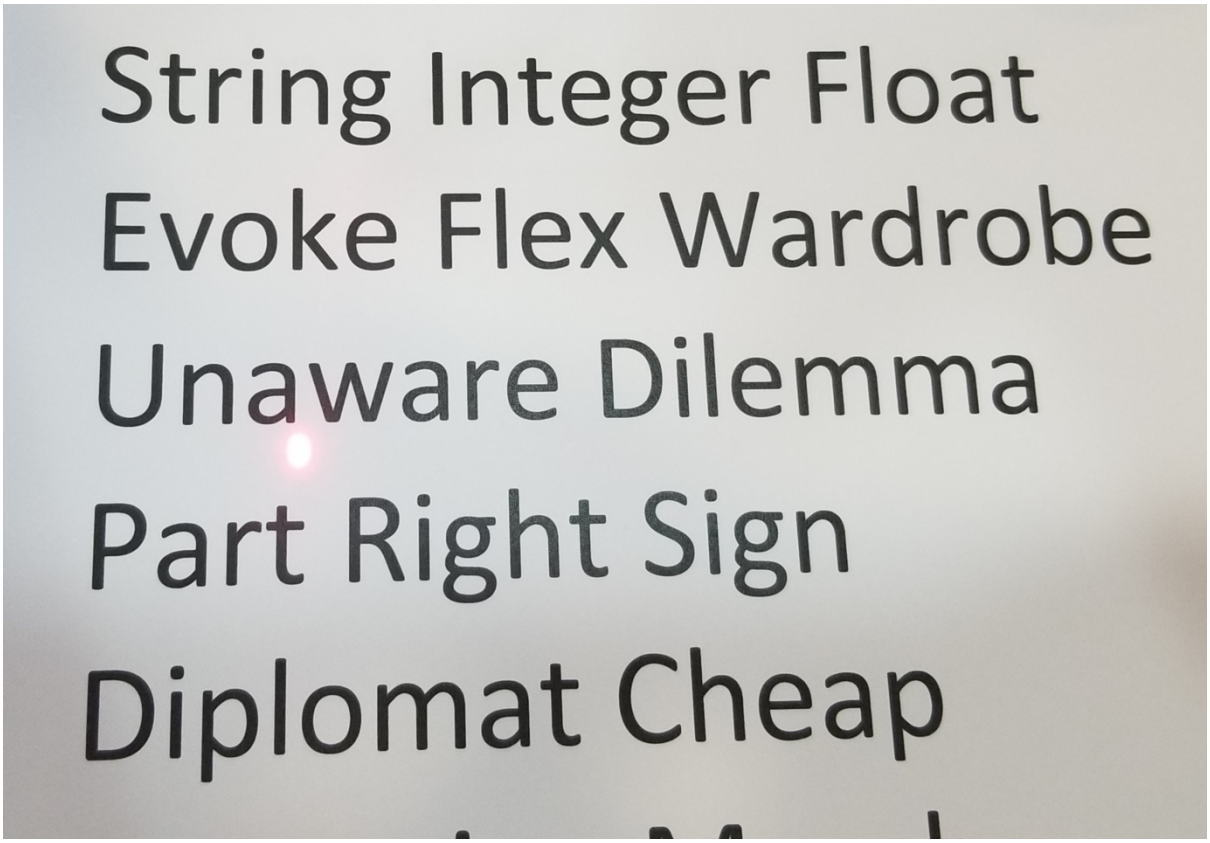
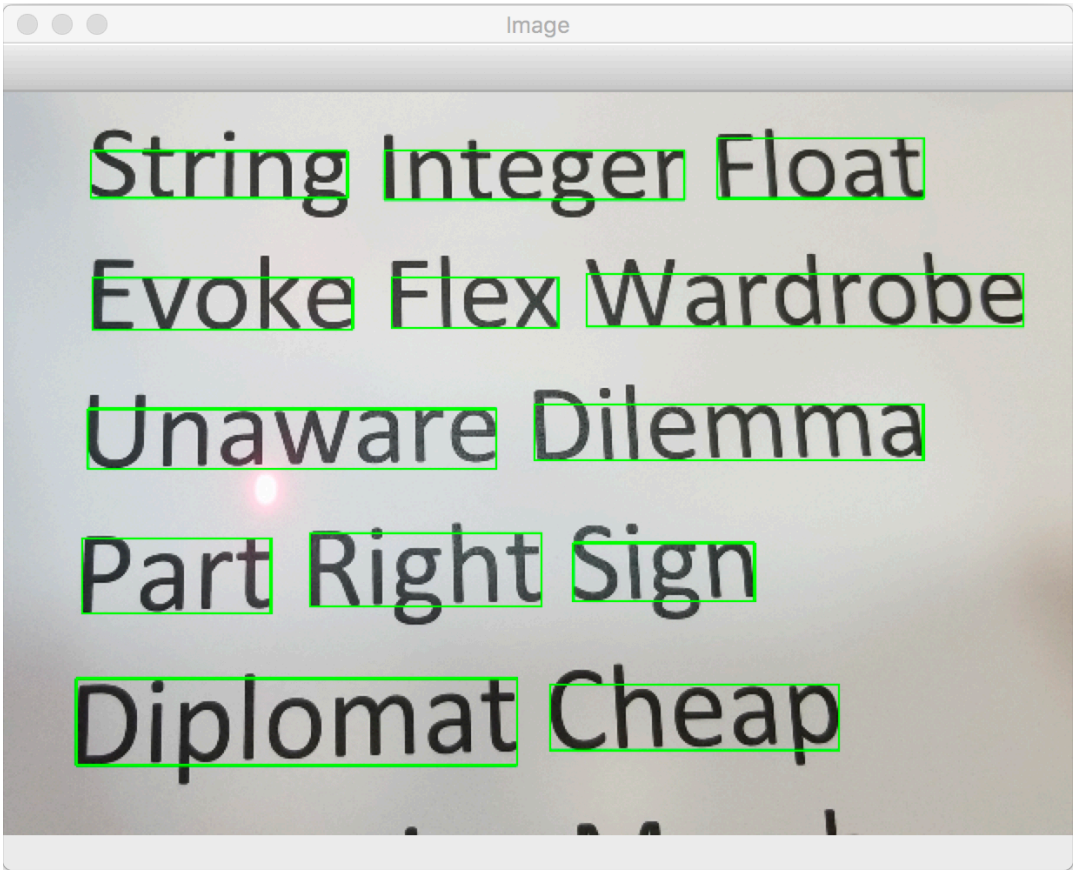
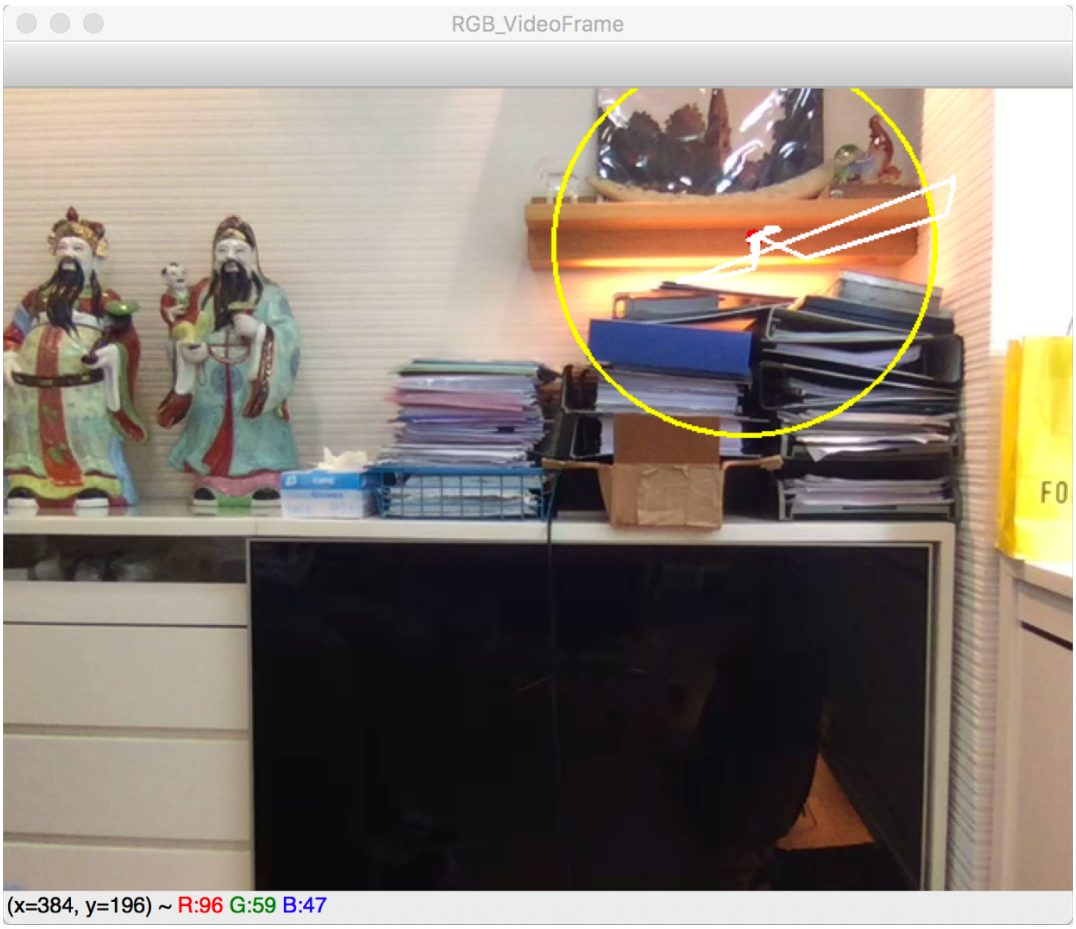


Figure 6



For the tracking of the laser dot, I used a program created by Brad Montgomery [7] [8]. This program runs very smoothly using my computer with no visible delay, but it does have its limitations. If the background you're working with is not a plain piece of paper it may provide a few anomalies as shown in Figure 7. Another issue is that even though there may not be a laser there, it mistakes certain lights in the background to be a laser. I have also tested that the tracker would only work up to a range of around 30cm.

Figure 7



After managing to create the program for finding and extracting words from a piece of paper and using Brad Montgomery's program for laser tracking, I managed to put my word extraction program into Montgomery's program. As Montgomery's program captures the video frame by frame and applying the laser tracker on each frame, I am able to apply the word extraction after the laser is found from each frame. After the word extraction program has been applied to the laser tracking program, you can see from the video feed output that the program is running significantly slower as the stream stutters when processing.

To overcome this problem, I decided to crop the frame before it is passed into the word extraction program. This is because I noticed that if there are less words in the image, the time it takes to run tesseract decreases dramatically. The results in Figure 8 shows the time difference between using the original image and using a cropped image. The results are collected using the image from Figure 5 and Figure 9. As you can see from the results, that by cropping the image the time it takes to process the picture decreases by about 75%.

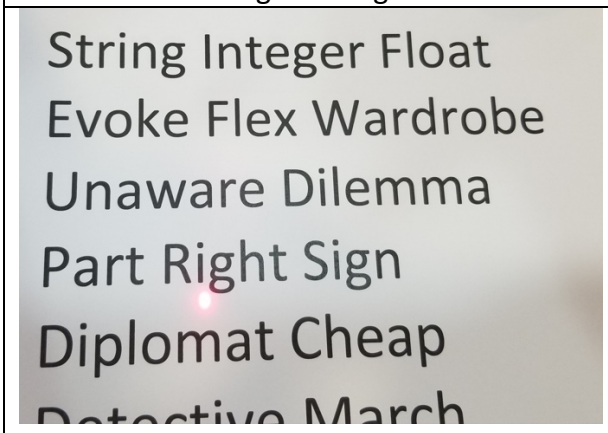
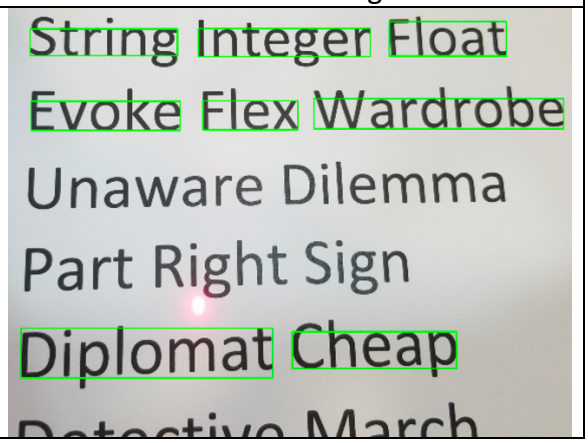
Figure 8

Figure 5 (s)	Figure 9 (s)
2.3776	0.5408

Figure 9

After merging the two programs together, another problem occurred. Due to all the words being on a single piece of paper, the laser may sometimes cover the word the user wants it to translate, in which case causing the word extraction part of the program to fail to operate correctly, as shown in Figure 10. As you can see, the red laser's light stops the program being able to detect the words around it and this may cause inaccuracy's when trying to extract a word. Furthermore, the fact the program just outputs all the words the laser comes across also contributes to this inaccuracy massively.

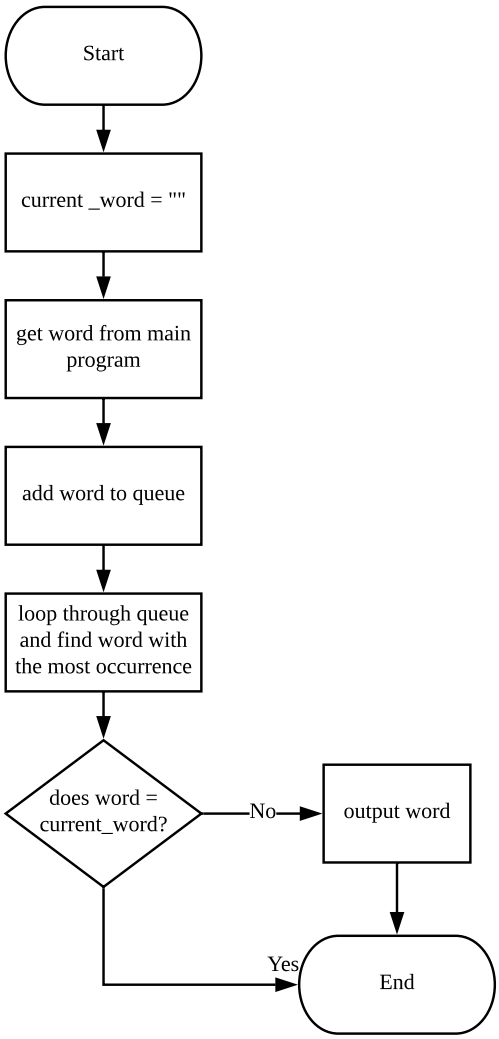
Figure 10

Original image	Processed image
	

In order to reduce this inaccuracy, I decided to use a queue which would store the last 20 words returned by the word extraction program. During each video frame iteration, it would count up the most frequently occurring word in that queue. If that word is not the same as the last word which was outputted by the program, then output that word. If it is the same, then continue. The diagram in **Error! Not a valid bookmark self-reference.** shows my solution to this problem.

I chose this solution to the problem because if the word extraction program manages to return the same word from multiple different video frames, it would suggest that that word is in fact the correct word that the user wants to be translated. Furthermore, you can also increase the 20 words threshold to further reduce the chance chance of an inaccurate word being produced, but this may cause the output of the word to take longer and so there needs to be a balance that should be found.

Figure 11



With the two main functions needed for my program to work, the only thing left is to translate and output the word through the speakers. In order to do this, I used googletrans to translate the word into the desired language, and then used the audio module to play the translated word. There is a problem with the googletrans module, this is that some words may not be translated correctly based on the context. For example, the word ‘cheap’ can have multiple meanings, it can mean that the cost of something is cheap, or somebody has a cheap personality, without context it is impossible for the googletrans module to translate correctly, which would cause incorrect translations.

Discussions

This project has certainly met the expectations I have made before I started. The program is able to extract a word based on where the laser pointer is pointing at, translate the word extracted and say it out with the speakers. Although expectations were met, there are still a few improvements that could be made.

The first improvement would be to increase the accuracy of the whole system, this includes the tracking of the laser, the extraction of text and the translation of the extracted text. Currently, background objects or lights may be mistaken as the laser. Because of this, when the laser is mistaken the word that you wanted to be extracted may be the wrong one causing the whole cycle to produce the wrong result. In the future, to try and decrease the chance of this error, maybe applying a different HSV filter may provide a better result.

There are also problems when extracting the text from the image. Assuming there are no problems when extracting the points of the laser, the image may sometimes be distorted or blurry which may cause errors when trying to extract text from that image as the text may not be viewable. This may cause the wrong text to be extracted, and hence the wrong translated text to be outputted. In the future, maybe using better hardware such as a camera that has optical image stabilisation, the image captured would be clearer reducing the chances of extraction error.

Furthermore, as stated before, the translation may translate to the wrong word if not given the proper context. To decrease the chance of this happening, in the future it may be preferable to translate the whole line instead of just one word. Although this may be able to reduce the chance of translating to the wrong word, by having to translate a whole sentence would require more time to process and hence reduce the efficiency of the system.

Finally, in the future I would like to decrease the time it takes for the whole text extraction to translation process, hopefully it may be able to translate words instantaneously. In order to achieve this, it may be better to write this program using C++. This is because Python is a language that uses an interpreter which is considerably slower than compiling a program before it is ran, which is what C++ does. Furthermore, OpenCV and Tesseract are both mainly written for C++, so the libraries may be more optimised for it.

Conclusion

The problem we have acquired at the start was that humans struggle to communicate when there is a language barrier. This causes a massive problem when we want to go abroad and visit other countries as that language barrier would most likely exist. With the current technology available to the majority of us, the language barrier should not be hard to overcome.

To summarise this research, I have created a program that is able to extract and translate words on the go. This application uses computer vision and OCR techniques to extract and make use of information from an image. There is still a lot of work required to put into this application to improve the quality of the translation service, but there is also a lot of development that could be made in the future. In the time ahead, we can implement other computational techniques like augmented reality and apply this application on something like the Google Glass. Hopefully, one day with this application everybody in the world would be able to communicate with each with ease no matter what language they can or cannot speak.

References

[1] R. Szeliski, “Computer Vision: Algorithms and Applications,” ISBN 978-1-84882-935-0, Springer Science & Business Media, 2010, pp. 10-16.

[2] E. E. F. d'Able, On a Type-Reading Optophone, 1 July 1914.

[3] L. Tatwany and H. C. Ouertani, “A review on using augmented reality in text translation,” IEEE, Muscat, Oman, 2017.

[4] M. Bargava, P. Dhote, A. Srivastava and A. Kumar, “Speech enabled integrated AR-based multimodal language translation,” IEEE, Pune, India, 2016.

[5] “newgenapps,” 29 May 2017. [Online]. Available: https://www.newgenapps.com/blog/python-for-ai-artificial-intelligence-ml. [Accessed 11 September 2018].

[6] Mindfire, “Medium,” 24 April 2017. [Online]. Available: https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121. [Accessed 11 September 2018].

[7] B. Montgomery, “bradmontgomery.net,” 24 January 2008. [Online]. Available: https://bradmontgomery.net/blog/tracking-a-laser-pointer-with-python-and-opencv/. [Accessed 11 September 2018].

[8] B. Montgomery, “Github,” 25 August 2016. [Online]. Available: https://github.com/bradmontgomery/python-laser-tracker. [Accessed 11 September 2018].

[9] S. Hoffstaetter, J. Bochi, M. Lee, L. Kistner, R. Mitchell and E. Cecchini, “Github,” [Online]. Available: https://github.com/madmaze/pytesseract. [Accessed 11 September 2018].